```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '85 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T85;
{ -- This program will simulate a Last-In-First-Out stack. }
  var
    A: String [4];
    S: Integer;
    N: Array [1..9] of Integer;

begin
  S := 0;
  repeat
    Write ('Enter command: ');  Readln (A);
    if A = 'ADD' then begin
      Inc(S);
      Write('Enter number: ');  Readln (N[S]);
    end;
    if A = 'TAKE' then begin
      Writeln (N[S]);  Dec(S);
    end;
  until A = 'QUIT';
end.



{1.2}
program One2T85;
{ -- This program will determine which number was erased. }
  var
    N, S, I, T: Integer;
    Av:         Real;

begin
  Write ('Enter N, AV: ');  Readln (N, Av);  S := 0;
  for I := 1 to N do S := S + I;
  for I := 1 to N do begin
    T := S - I;
    if T / (N - 1) = Av then begin
      Writeln ('NUMBER ERASED WAS ', I);  Exit;
    end;
  end;
end.
```

```
{1.3}
program One3T85;
{ -- This program will print the square root of N. }
  var
    D, I, T, V, Code: Integer;
    N, Q, S, Pow:     Real;
    A:                String[9];
    C:                Char;

begin
  Write ('Enter N, D: ');  Readln (N, D);
  Q := Sqrt(N);  T := 0;
  Pow := 1;
  for I := 1 to Abs(D) do Pow := Pow * 10;
  if D < 0 then Pow := 1 / Pow;
  S := Int (Q / Pow + 0.5) * Pow;
  Str (S: 4:4, A);
  for I := 1 to Length(A) do begin
    C := A[I];
    if C <> '.' then begin
      Val(C, V, Code);  T := T + V;
    end;
  end;
  Writeln ('S=', S :9:4);
  Writeln ('SUM=', T :3);
end.
```

```
{1.4}
program One4T85;
{ -- This program will simulate a time dial. }
uses Crt;
  var
    Y, J, K: Integer;

begin
  ClrScr;  Y := 1985;  J := 661;
  while Y <= 2345 do begin
    GotoXY (38,12);  Write (Y);
    if J > 10 then Dec(J,10);
    Delay (J);
    Inc(Y);
  end;
end.
```

```
{1.5}
program One5T85;
{ -- This program will determine # of tennis games and byes. }
  var
    N, G, B, R, TG, BY: Integer;

begin
  Write ('Enter N: ');  Readln (N);
  R := 0;  TG := 0;  BY := 0;
  while N > 1 do begin
    G := N div 2;
    if G * 2 = N then B := 0 else B := 1;
    Inc(R);  Write ('ROUND ', R, '  ', G:2, ' GAMES');
    if B = 1 then
      Writeln ('  1 BYE')
    else
      Writeln;
    TG := TG + G;  BY := BY + B;  N := G + B;
  end;
  Writeln ('TOTAL    ', TG:2, ' GAMES  ', BY, ' BYES');
end.
```

```
{1.6}
program One6T85;
{ -- This program will find smallest, largest and sum of #s. }
  var
    N, M, I, H, Num, T, U, L: Integer;
    S: LongInt;

begin
  Write ('Enter N, M: ');  Readln (N, M);  S := 0;
  if M > 999 then M := 999;
  if N < 100 then N := 100;
  for I := N to M do begin
    Num := I;
    H := Num div 100;  Num := Num - H * 100;
    T := Num div 10;     U := Num - T * 10;
    if (T = 0) or (U = 0) or (H = T) or (T = U) or (H = U) then
    else begin
      S := S + I;  L := I;
      if S = I then Writeln ('SMALLEST = ', I);
    end;
  end;
  Writeln ('LARGEST = ', L);
  Writeln ('SUM = ', S);
end.
```

```
{1.7}
program One7T85;
{ -- This program will print a bill for Bob's Cycle shop. }
  const
    A: Array [1..7] of String[4] =
      ('S193', 'S867', 'F234', 'S445', 'C492', 'J273', 'T100');
    B: Array [1..7] of String[20] =
      ('10 INCH SPROCKET', '30 INCH CHAIN', 'BLITZ MAG FRAME',
       'COMPUTCYCLE COMPUTER', 'JET BRAKE SET', '27 INCH WHEEL',
       '27X1 INCH TIRE TUBE');
    C: Array [1..7] of Real =
      (13.95, 27.50, 119.00, 33.95, 29.98, 32.00, 12.50);
  var
    N, P:             String[10];
    I:                Integer;
    LT, LC, Tot, Tax: Real;

begin
  Write ('Enter Customer name: ');  Readln (N);
  Write ('Enter part#: ');  Readln (P);
  Write ('Enter labor time: ');  Readln (LT);
  I := 1;
  while (P <> A[I]) and (I < 7) do Inc(I);
  Writeln ('CUSTOMER NAME: ', N);
  Writeln ('PART #: ', P);
  Writeln ('DESCRIPTION: ', B[I]);
  Writeln ('PART COST:  ', C[I]: 6:2);
  LC := LT * 10;
  Writeln ('LABOR COST: ', LC: 6:2);
  Tax := C[I] * 0.05;
  Tax := Int(Tax * 100.0 + 0.501) / 100.0;
  Writeln ('5% TAX:     ', Tax :6:2);
  Tot := LC + C[I] + Tax;
  Writeln ('TOTAL:      ', Int(Tot * 100 + 0.5) / 100 :6:2);
end.
```

```
{1.8}
program One8T85;
{ -- This program will display labels alphabetically. }
  const
    A: Array [1..6] of String[16] = ('LISA SPINXS', 'BOB SIMON',
      'BILL SIMON', 'HARRY TROUTMAN', 'HARRY PARKER', '*END*');
    B: Array [1..6] of String[8] =  ('987-6543', '923-4455',
      '123-4567', '876-2174', '222-3333', '0');
  var
    H, S, L, I, J: Integer;
    Rst, Lst:      String[10];
    X:             String[18];
    C:             Array [1..6] of String[18];

begin
  Write ('Enter # of lines on label: ');  Readln (H);
  S := 1;
  while A[S] <> '*END*' do begin
    L := Length(A[S]);  I := 1;
    while Copy(A[S], I, 1) <> ' ' do Inc(I);
    Rst := Copy(A[S], I+1, L-I);  Lst := Copy (A[S], 1, I);
    C[S] := Rst + ', ' + Lst;
    Inc(S);
  end;
  Dec(S);
  for I := 1 to S - 1 do
    for J := I+1 to S do
      if C[I] > C[J] then begin
        X := C[I];  C[I] := C[J];  C[J] := X;
        X := B[I];  B[I] := B[J];  B[J] := X;
      end;
  for I := 1 to S do begin
    Writeln;  Writeln (C[I]);  Writeln (B[I]);
    for J := 1 to H - 3 do Writeln;
  end;
end.
```

```
{1.9}
program One9T85;
{ -- This program will guess secret letter in 5x5 matrix. }
uses Crt;
  var
    I, J, S, X: Integer;
    C: Char;
    A: Array [0..24] of Integer;
    B: Array [1..5, 1..5] of Char;

begin
   ClrScr;  Randomize;  S := 11;
   for I := 0 to 24 do A[I] := 0;
   for I := 1 to 5 do
     for J := 1 to 5 do begin
       repeat
         X := Random(25);
       until A[X] = 0;
       B[I, J] := Chr(X + 65);
       GotoXY (13 + J * 2, I);  Write (B[I, J]);  A[X] := 1;
     end;

  I := 0; C := ' ';
  while (C <> 'Y') and (S > 0) do begin
    GotoXY (30, 2);  Write ('SCORE=', S:2);  Dec(S);;
    GotoXY (10, 10);  Inc(I);
    Write ('IS THE LETTER IN ROW ', I, ' ');  Readln (C);
  end;
  J := 0;  C := ' ';
  while (C <> 'Y') and (S > 0) do begin
    GotoXY (30, 2);  Write ('SCORE=', S:2);  Dec(S);
    GotoXY (10, 12);  Inc(J);
    Write ('IS THE LETTER IN COL ', J, ' ');  Readln (C);
  end;
  if S > 0 then Writeln ('YOUR LETTER IS ', B[I,J]);
end.
```

```
{1.10}
program One10T85;
{ -- This program will display squares relative to cursor and #.
}
uses Crt;
  var
    R, C, X, A, B: Integer;
    K: char;

begin
  ClrScr;  R := 5;  C := 5;  K := ' ';
  while not (K in ['1' .. '4']) do begin
    GotoXY (C, R);  Write ('#');  K := ' ';
    K := ReadKey;
    if K in ['I', 'J', 'K', 'M'] then begin
      GotoXY (C, R);  Write (' ');
      if K = 'I' then Dec(R);
      if K = 'M' then Inc(R);
      if K = 'J' then Dec(C);
      if K = 'K' then Inc(C);
      K := '5';
    end;
  end;
  X := Ord(K) - Ord('0');
  if X = 1 then begin  A := 1;  B :=  0;  end;
  if X = 2 then begin  A := 1;  B := -1;  end;
  if X = 3 then begin  A := -1; B := -1;  end;
  if X = 4 then begin  A := -1; B := 0;   end;
  if (R + 5*A > 24) or (R + 5*A < 1) or
     (C + 9*B > 80) or (C + 9*B < 1) then
       Writeln ('OFF THE SCREEN')
  else begin
    GotoXY (C + 8*B, R + 1*A);  Writeln ('*********');
    GotoXY (C + 8*B, R + 2*A);  Writeln ('*       *');
    GotoXY (C + 8*B, R + 3*A);  Writeln ('*   ', X, '   *');
    GotoXY (C + 8*B, R + 4*A);  Writeln ('*       *');
    GotoXY (C + 8*B, R + 5*A);  Writeln ('*********');
  end;
end.
```

```
{2.1}
program Two1T85;
{ -- This program will outline screen with random letters. }
uses Crt;
  var
    I, J, X: Integer;
    A, Ch:   Char;

begin
  repeat
    Randomize;  ClrScr;
    for I := 1 to 11 do begin
      X := Random(26);  A := Chr(65 + X);
      GotoXY (I, I);
      for J := I to 80 - I do Write (A);
      for J := I+1 to 23-I do begin
        GotoXY (I, J);     Write (A);
        GotoXY (80-I, J);  Write (A);
      end;
      GotoXY (I, 23-I);
      for J := I to 80 - I do Write (A);
      Ch := ReadKey;
    end;
  until Ch = Chr(27);
  ClrScr;
end.
```

```
{2.2}
program Two2T85;
{ -- This program will print the longest sequence of letters. }
  var
    N, I, J, K: Integer;
    A:          Array [1..20] of Char;
    Found, One: Boolean;

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter letter: ');  Readln (A[I]);
  end;
  I := N;  Found := False;
  while (I >= 2) and not Found do begin
    for J := 1 to N-I+1 do begin
      One := True;
      for K := 0 to I-2 do
        if A[J+K] >= A[J+K+1] then One := False;
      if One then begin
        for K := 0 to I-1 do Write (A[J+K], ' ');
        Writeln;  Found := True;
      end;
    end;
    Dec(I);
  end;
end.
```

```
{2.3}
program Two3T85;
{ -- This program will change the margins for a given text. }
  var
    A: String[128];
    W: String[20];
    C: Char;
    I, L, LW, LL: Integer;

begin
  Write ('Enter text: ');  Readln (A);  A := A + ' ';
  L := Length(A);  LW := 5;  Write (' ': 10);  W := '';
  for I := 1 to L do begin
    C := A[I];
    if C <> ' ' then
      W := W + C
    else begin
      LL := Length(W);
      if LW + LL > 30 then begin
        Writeln;  Write (' ': 5);  LW := 0;
      end;
      if LL > 0 then begin
        Write (W, ' ');  LW := LW + LL + 1;  W := '';
      end;
      if (LL = 0) and (LW > 0) then begin
        Write (' ');  Inc(LW);
      end;
    end;
  end;
end.
```

```
{2.4}
program Two4T85;
{ -- This program will print word with consonants alphabetized. }
  const
    Vowels: String[5] = 'AEIOU';
  var
    I, J, L, VV, CC, VN, CN: Integer;
    A: String[20];
    B, X: Char;
    C: Array [1..20] of Char;
    V: Array [1..20] of Char;
    D: Array [1..20] of Char;

begin
  Write ('Enter word: ');  Readln (A);  L := Length(A);
  CN := 0;  VN := 0;  CC := 0;  VV := 0;
  for I := 1 to L do begin
    B := A[I];  J := 1;
    while (J < 5) and (Copy(Vowels, J, 1) <> B) do Inc(J);
    if Copy (Vowels, J, 1) <> B then begin
      Inc(CN);  C[CN] := B;  D[I] := 'C';  end
    else begin
      Inc(VN);  V[VN] := B;  D[I] := 'V';
    end;
  end;

  { -- Sort Vowels }
  for I := 1 to VN-1 do
    for J := I+1 to VN do
      if V[I] > V[J] then begin
        X := V[I];  V[I] := V[J];  V[J] := X;
      end;

  { -- Sort Consonants }
  for I := 1 to CN-1 do
    for J := I+1 to CN do
      if C[I] > C[J] then begin
        X := C[I];  C[I] := C[J];  C[J] := X;
      end;

  for I := 1 to L do
    if D[I] = 'V' then begin
      Inc(VV);  Write (V[VV]);  end
    else begin
      Inc(CC);  Write (C[CC]);
    end;
  Writeln;
end.
```

```
{2.5}
program Two5T85;
{ -- This program will print common letters and line up words. }
  var
    N, I, J, K:    Integer;
    Common, Found: Boolean;
    X, Let:        Char;
    A:             Array [1..10] of String[15];

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter word: ');  Readln (A[I]);
  end;
  Found := False;
  for I := 1 to 26 do begin
    X := Chr(64 + I);  Common := True;  J := 1;
    while (J <= N) and Common do begin
      K := 1;
      while (K <= Length(A[J])) and (Copy(A[J], K, 1) <> X) do
        Inc(K);
      if Copy(A[J], K, 1) <> X then Common := False;
      Inc(J);
    end;
    if Common then begin
      Write (X, '  ');  Found := True;
    end;
  end;
  if not found then begin
    Writeln ('NO COMMON LETTERS');  Exit;
  end;
  Writeln;  Write ('Choose letter: ');  Readln (Let);
  for I := 1 to N do begin
    J := 1;
    while (Copy(A[I], J, 1) <> Let) do Inc(J);
    Writeln (' ': 10 - J, A[I]);
  end;
end.
```

```
{2.6}
program Two6T85;
{ -- This program will keep score for a double dual race. }
  var
    Init:                Array [1..21] of String[2];
    TeamName:            Array [1..3] of String[2];
    I, J, K:             Integer;
    StillUnique:         Boolean;
    UniqueTeams, Pl:     Integer;
    Team1Pos, Team2Pos: Array [1..7] of Integer;
    Team1,    Team2:     Integer;
    Team1Pl,  Team2Pl:   Integer;
```

```
begin
  UniqueTeams := 0;
  for I := 1 to 21 do begin
    Write ('Place ', I: 2, ': ');  Readln (Init[I]);
    J := 0;  StillUnique := True;
    while (J < UniqueTeams) and StillUnique and (I > 1) do begin
      Inc(J);
      if TeamName[J] = Init[I] then
        StillUnique := False;
    end; { -- while }
    if StillUnique then
      begin
        Inc(UniqueTeams);
        TeamName[UniqueTeams] := Init[I];
      end;
  end; { -- for I }
  { -- Assert that Team[1,2,3] = 3 unique team Initials. }

  for I := 1 to 2 do
    for J := I+1 to 3 do begin
      PL := 0;  Team1 := 0;  Team2 := 0;
      Team1Pl := 0;  Team2Pl :=0;
      for K := 1 to 21 do begin
        if Init[K] = TeamName[I] then
        begin
          Inc(Pl);
          Team1 := Team1 + Pl;
          Inc(Team1Pl);
          Team1Pos[Team1Pl] := Pl
        end;
        if Init[K] = TeamName[J] then
        begin
          Inc(Pl);
          Team2 := Team2 + Pl;
          Inc(Team2Pl);
          Team2Pos[Team2Pl] := Pl
        end;
      end;  { -- for K }
      Team1 := Team1 - Team1Pos[6] - Team1Pos[7];
      Team2 := Team2 - Team2Pos[6] - Team2Pos[7];
      Writeln ('TEAM ', TeamName[I], ': ', Team1, ' POINTS');
      Writeln ('TEAM ', TeamName[J], ': ', Team2, ' POINTS');
      if (Team1 < Team2)
      or ((Team1 = Team2) and (Team1Pos[6] < Team2Pos[6])) then
        Write ('TEAM ', TeamName[I])
      else
        Write ('TEAM ', TeamName[J]);
      Writeln (' WINS!');  Writeln;
    end;  { -- for J }
end.
```

```
{2.7}
program Two7T85;
{ -- This program will allow manipulation of 3x3 array of data. }
uses Crt;
  var
    A: Array [1..4, 1..4] of Real;
    Tot:            Real;
    I, J, Row, Col: Integer;
    C, Ch:          Char;

begin
  A[1,1] := 10.11;  A[1,2] := 20.22;  A[1,3] := 30.33;
  A[2,1] := 11.1;   A[2,2] := 22.2;   A[2,3] := 33.3;
  A[3,1] := 10.0;   A[3,2] := 20.0;   A[3,3] := 30.0;
  C := ' ';
  while C <> 'C' do begin
    ClrScr;
    Writeln ('A. EDIT OR CHANGE A VALUE');
    Writeln ('B. DISPLAY THE RESULTS');
    Writeln ('C. QUIT');
    Write ('Enter option: ');  Readln (C);
    if C = 'A' then begin
      Write ('Enter row, col: ');  Readln (Row, Col);
      Write ('Enter number: ');    Readln (A[Row, Col]);
      end
    else if C = 'B' then begin
      for I := 1 to 3 do A[I, 4] := 0;
      for J := 1 to 3 do A[4, J] := 0;
      Tot := 0;
      for I := 1 to 3 do begin
        for J := 1 to 3 do begin
          Write (A[I,J] :6:2, '  ');  Tot := Tot + A[I, J];
          A[4, J] := A[4, J] + A[I, J];
          A[I, 4] := A[I, 4] + A[I, J];
        end;
        Writeln (A[I, 4]: 6:2);
      end;
      for J := 1 to 3 do Write (A[4, J] :6:2, '  ');
      Write (Tot :6:2);
    end;
    if C <> 'C' then begin
      Writeln;  Write ('Press any key: ');  Ch := ReadKey;
    end;
  end;  { -- while }
end.
```

```
{2.8}
program Two8T85;
{ -- This program will print all combinations of 4 digits. }
  var
    A, B, C, D, P, S, Code: Integer;
    Pst: String[2];

begin
  S := 0;
  for A := 1 to 8 do
    for B := A+1 to 9 do begin
      P := A * B;
      if P >= 10 then begin
        Str(P, Pst);
        Val(Copy(Pst,1,1), C, Code);
        Val(Copy(Pst,2,1), D, Code);
        if (A <> C) and (A <> D) and (B <> C) and (B <> D) then
        begin
          Write   (A, ' ', B, ' ', C, ' ', D, '   ');
          Writeln (A, ' X ', B, ' = ', P);
          Inc(S);
        end;
      end;
    end;
  Writeln ('TOTAL = ', S);
end.
```

```
{2.9}
program Two9T85;
{ -- This program will select words given a string w/ wildcard. }
  var
    A:              Array[1..25] of String[11];
    I, J, N, L, W: Integer;
    St, X, Ri, Le: String[11];

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter word: ');  Readln (A[I]);
  end;
  repeat
    Write ('Enter string: ');  Readln (St);
    L := Length(St);  W := 0;  X := '';
    I := Pos('*', St);
    if I = 0 then Exit;
    { -- Asterisk is at position I }
    { -- Compare Left part of string and Right part of string. }
    Le := Copy(St, 1, I-1);  Ri := Copy (St, I+1, L-I);
    for J := 1 to N do
      if (Copy(A[J], 1, I-1) = Le) and
         (Copy(A[J], Length(A[J]) - (L-I) + 1, L-I) = Ri) then
        begin
          Writeln (A[J]);  W := 1;
        end;

    if W = 0 then Writeln ('NO WORDS FOUND');  Writeln;
  until I = 0;
end.
```

```
{2.10}
program Two10T85;
{ -- This program will maintain air conditioning in 3 rooms. }
uses Crt;
  var
    Off, Co, Dr:                Real;
    S, M, O, C, D, Ch, Air, LM: Integer;
    OfAir, CoAir, DrAir:        Real;

begin
  Write ('Enter last 5-minutes: '); Readln (LM);
  ClrScr;
  Off   := 72;    Co := 65;    Dr := 79;
  OfAir := 0;  CoAir := 0;  DrAir := 0;
      S := 0;      M := 0;     Ch := 0;
      O := 0;      C := 0;      D := 0;
  Writeln ('OF CO DS  OFFICE  COMP.   DRY.   MIN:SEC');
  repeat
    if ((M mod 5 = 0) and (S = 0)) or (Ch = 1) then begin
      Write (O, '  ', C, '  ', D, '   ');
      Write (Off: 3:1, '    ', Co :3:1, '    ', Dr :3:1);
      Write ('   ', M:3, ':');
      if S > 0 then Writeln (S) else Writeln ('00');
      Ch := 0;
    end;
    S := S + 15;
    if S = 60 then begin
      Inc(M);  S := 0;
    end;
    Off := Off + 0.1   - OfAir;
    Co  := Co  + 0.2   - CoAir;
    Dr  := Dr  + 0.1/4 - DrAir;
    if (Off > 78) and (O = 0) then begin O := 1;  Ch := 1;  end;
    if (Co > 70)  and (C = 0) then begin C := 1;  Ch := 1;  end;
    if (Dr > 85)  and (D = 0) then begin D := 1;  Ch := 1;  end;
    if (Off < 72) and (O = 1) then begin O := 0;  Ch := 1;  end;
    if (Co < 65)  and (C = 1) then begin C := 0;  Ch := 1;  end;
    if (Dr < 75)  and (D = 1) then begin D := 0;  Ch := 1;  end;
    Air := (O + C + D) * 2;
    if Air = 0 then begin
      OfAir := 0;  CoAir := 0;  DrAir := 0;  end
    else begin
      OfAir := O / Air;  CoAir := C / Air;  DrAir := D / Air;
    end;
  until (M = LM) and (S > 0);
end.
```

```
{3.1}
program Thr1T85;
{ -- This program will display the sides of a die. }
  { -- 6 ways to represent die (each with different top)
       DATA Top, Front, Right, Back, Left, (Bottom derived) }
  const
    A: Array[1..30] of Integer =
      (1, 5, 4, 2, 3,  6, 5, 3, 2, 4,  5, 1, 3, 6, 4,
       2, 1, 4, 6, 3,  3, 5, 1, 2, 6,  4, 5, 6, 2, 1);
  var
    T, F, I, J, R: Integer;

begin
  Write ('Enter Top, Front: ');  Readln (T, F);
  { -- Determine which data set of 5 to use (based on top #) }
  I := 1;
  while A[I] <> T do I := I + 5;
  { -- Rotate sides till a side matches the front # }
  J := 1;
  while (A[I + J] <> F) do J := J + 1;
  if J = 4 then J := 0;
  R := J + 1;
  { -- Generate rest of sides, sum of opposites sides = 7 }
  Writeln ('TOP = ', T, '  FRONT = ', F, '  RIGHT = ', A[I+R]);
  Write   ('BACK = ', 7-F, '  LEFT = ', 7 - A[I+R]);
  Writeln ('  BOTTOM = ', 7-T);
end.
```

```
{3.2}
program Thr2T85;
{ -- This program will factor a quadratic equation. }
  var
    A, B, C, D, E, H, I, K, N, S: Integer;
    R: Array [1..2] of Integer;
    Displayed: Boolean;

begin
  Write ('Enter A, B, C: ');  Readln (A, B, C);
  if A < 0 then begin
    A := -A;  B := -B;  C := -C;
  end;
  if A > 1 then
    for I := A downto 2 do
      if (A mod I = 0) and (B mod I = 0) and (C mod I = 0) then
      begin
        A := A div I;  B := B div I;  C := C div I;  Write (I);
      end;

  S := B * B - 4 * A * C;
  if S < 0 then begin
    Writeln ('CANNOT BE FACTORED'); Exit;
  end;
  H := Trunc (Sqrt(S) + 0.01);  E := 2 * A;
  R[1] := -B + H;  R[2] := -B - H;
  for K := 1 to 2 do begin
    D := E;  N := R[K];  I := D;  Displayed := False;
    repeat
      if (N mod I = 0) and (D mod I = 0) then begin
        N := N div I;  D := D div I;
        Write ('(');
        if D > 1 then Write (D);
        Write ('X');
        if N < 0 then Write ('+', (-N), ')');
        if N > 0 then Write ('-', N, ')');
        Displayed := True;
      end;
      Dec(I);
    until Displayed;
  end;
end.
```

```
{3.3}
program Thr3T85;
{ -- This program will simulate a calculator. }
  var
    I, J, K, L, Code: Integer;
    Ex, C: String[20];
    Ch:    String[1];
    S:     Real;
    B:     Array [1..10] of Integer;
    A:     Array [1..10] of Real;

begin
  Write ('Enter expression: ');  Readln (Ex);
  L := Length(Ex);  C := '';  J := 0;
  for I := 1 to L do begin
    Ch := Copy (Ex, I, 1);
    if Ch >= '0' then
      C := C + Ch
    else begin
      Inc(J);  Val(C, A[J], Code);  C := '';
      B[J] := Pos(Ch, '+-*/');
    end;
  end;
  Inc(J);  Val(C, A[J], Code);  K := 1;
  for I := 1 to J-1 do
    if B[I] < 3 then begin
      B[K] := B[I];  Inc(K);  A[K] := A[I+1];  end
    else
      if B[I] = 3 then
        A[K] := A[K] * A[I+1]
      else  { -- B = 4 }
        A[K] := A[K] / A[I+1];

  S := A[1];
  for I := 1 to K-1 do
    if B[I] = 2 then S := S - A[I+1] else S := S + A[I+1];
  Writeln (S: 7:3);
end.
```

```
{3.4}
program Thr4T85;
{ -- This program will compute all digits of N factorial. }
  var
    N, I, J, D, C, CC: Integer;
    A: Array [1..254] of Integer;

begin
  Write ('Enter N: ');  Readln (N);
  D := 1;  A[1] := 1;  C := 0;
  for I := 1 to N do begin
    for J := 1 to D do begin
      A[J] := A[J] * I + C;  C := A[J] div 10;
      A[J] := A[J] - 10 * C;
    end;
    while C > 0 do begin
      CC := C div 10;  Inc(D);  A[D] := C - 10 * CC;  C := CC;
    end;
  end;
  for I := D downto 1 do Write (A[I]);
end.
```

```
{3.5}
program Thr5T85;
{ -- This program will sum and subtract 2 big decimals. }
  var
    Ast, Bst:   String[31];
    A, B, C, D: Array [1..30] of Integer;
    I, J, LenA, LenB, X, S, G,
    H, Y, Z, L, Code, Car, Bor:  Integer;

begin
  Write ('Enter #1: ');  Readln (Ast);  LenA := Length(Ast);
  Write ('Enter #2: ');  Readln (Bst);  LenB := Length(Bst);
  S := 0;
  for I := LenA downto 1 do
    if Copy(Ast, I, 1) = '.' then
      X := I
    else begin
      Inc(S);  Val(Copy(Ast, I, 1), A[S], Code);
    end;
  S := 0;
  for I := LenB downto 1 do
    if Copy (Bst, I, 1) = '.' then
      Y := I
    else begin
      Inc(S);  Val(Copy(Bst, I, 1), B[S], Code);
    end;
```

```
  { -- Allign decimal point }
  G := LenA - X;  H := LenB - Y;
  if G > H then L := G else L := H;
  Z := G - H;
  if Z > 0 then  { -- Second # is smaller, so place leading 0s. }
    begin
      for I := LenB-1 downto 1 do begin
        B[I+Z] := B[I];  B[I] := 0;
      end;
      LenB := LenB + Z;
    end;
  if Z < 0 then  { -- First # is smaller, so put leading 0s. }
    begin
      for I := LenA-1 downto 1 do begin
        A[I-Z] := A[I];  A[I] := 0;
      end;
      LenA := LenA - Z;
    end;

  if LenA > LenB then Y := LenA - 1 else Y := LenB - 1;
  Car := 0;  Bor := 0;
  { -- Add and subtract }
  for I := 1 to Y do begin
    C[I] := A[I] + B[I] + Car;  Car := C[I] div 10;
    C[I] := C[I] - Car * 10;
    D[I] := A[I] - B[I] - Bor;
    if D[I] < 0 then Bor := 1 else Bor := 0;
    D[I] := D[I] + Bor * 10;
  end;
  Write ('SUM = ');
  if Car > 0 then Write (Car);
  for I := Y downto 1 do begin
    if I = L then Write ('.');
    Write (C[I]);
  end;
  Writeln;  Write ('DIFFERENCE = ');
  for I := Y downto 1 do begin
    if I = L then Write ('.');
    Write (D[I]);
  end;
end.


{3.6}
program Thr6T85;
{ -- This program will control the movements of a snake. }
uses Crt;
  const
    SnakeLen = 30;
  var
    V, H, I, X, Y:  Integer;
    VCoord, HCoord: Array [1..SnakeLen] of Integer;
    FrontHV, EndHV: Integer;
    Ch:             Char;
    InvalidKey:     Boolean;
```

```
begin
  ClrScr;
  InvalidKey := False;
  V := 12;  H := 40-(SnakeLen div 2);  GotoXY (H,V);
  FrontHV := 0;   EndHV := 1;
  { -- Center snake (asterisks) on the screen }
  for I := H to (H + SnakeLen - 1) do begin
    Write ('*');
    Inc(FrontHV);
    VCoord[FrontHV] := V;
    HCoord[FrontHV] := I;
  end;
  Ch := ReadKey;

  repeat
    H := HCoord[FrontHV];
    V := VCoord[FrontHV];
    for I := 1 to 2000 do
      If KeyPressed then Ch := ReadKey;

    case Ch of
      'I', 'i' :  Dec(V);
      'M', 'm' :  Inc(V);
      'J', 'j' :  Dec(H);
      'K', 'k' :  Inc(H);
    end;

    for I := 1 to SnakeLen do
      if (H = HCoord[I]) and (V = VCoord[I]) then
        InValidKey := True;

    if InValidKey or (V = 0) or (V = 25) or (H = 0) or (H = 80)
then
      InvalidKey := True
    else begin
      GotoXY (H,V);  Write ('*');
      Y := HCoord[EndHV];
      X := VCoord[EndHV];
      GotoXY (Y,X);  Write (' ');
      HCoord[EndHV] := H;
      VCoord[EndHV] := V;
      Inc(FrontHV);
      if FrontHV > SnakeLen then
        FrontHV := 1;
      Inc(EndHV);
      If EndHV > SnakeLen then
        EndHV := 1;
    end; { -- else }
  until InvalidKey;
end.
```

```
{3.7}
program Thr7T85;
{ -- This program will print 3 permutations of a word. }
  var
    A:   String[8];
    Let: Char;
    F:   Array [1..7] of Integer;
    B:   Array [1..7] of Byte;
    KK, L, I, J, Fac, T, S, K, X: Integer;

begin
  Write ('Enter word: ');  Readln (A);  L := Length(A);
  Write ('Enter K: ');     Readln (KK);
  { -- Alphabetize letters }
  for I := 1 to L-1 do
    for J := I+1 to L do
      if A[I] > A[J] then begin
        Let := A[I];  A[I] := A[J];  A[J] := Let;
      end;

  { -- Produce factorials F(I) = (I-1)! }
  for I := 1 to L do begin
    Fac := 1;
    for J := 1 to I-1 do Fac := Fac * J;
    F[I] := Fac;
  end;

  for T := 1 to 3 do begin
    K := KK * T - 1;
    for I := 1 to L do B[I] := 0;
    { -- Generate Kth permuation }
    for I := L downto 1 do begin
      X := K div F[I];  S := 0;  J := 1;
      repeat
        while B[J] > 0 do Inc(J);
        Inc(S);
        if S > X then begin
          B[J] := 1;  Write (A[J]);  end
        else
          Inc(J);
      until (J > L) or (S > X);
      K := K - F[I] * X;
    end;
    Write ('  ');
  end;
  Writeln;
end.
```

```
{3.8}
program Thr8T85;
{ -- This program will display N pennies on board. }
uses Crt;
  var
    N, Sp, J, S, I:      Integer;
    A:  Array [1..14] of Integer;
    Ch: Char;

begin
  Write ('Enter N: ');  Readln (N);
  ClrScr;  Writeln ('TOTAL = ', N);
  if N = 8 then Sp := 1;  { -- 8 and 14 are special cases }
  J := N mod 2;  J := 2 - J;  S := J;
  if N = 14 then S := J + 2;
  Write (' ');
  for I := 1 to N do begin
    Write (I mod 10 :2);
  end;
  Writeln;
  for I := 1 to N do Writeln (I mod 10);
  for I := 1 to N do begin
    A[I] := S;
    if (N = 14) and (I = 14) then begin
      S := 2;  A[I] := S;
    end;
    GotoXY (2*S+1, 2+I);  Write ('*');
    S := S + 2 + Sp;
    if S > N then
      if (Sp = 1) then S := S - N else S := (N mod 2) + 1;
  end;
  Ch := ReadKey;
  for I := 1 to N do begin
    GotoXY (45, I+2);  Write ('(', I, ',', A[I], ')');
    Writeln ('   SUM = ', I + A[I]);
  end;
end.
```

```
{3.9}
program Thr9T85;
{ -- This program will determine # of moves made to a stack. }
  var
    N, I: Integer;
    A: Array [1..15] of Integer;

begin
  { 1 block - 1 move  (obvious)
    2 blocks- 3 moves (Move 1 stack, move #2, move 1 stack)
    3 blocks- 7 moves (Move 2 stack, move #3, move 2 stack on #3)
                      (3 moves     + 1 move +  3 moves)
    4 blocks-15 moves (Move 3 stack, move #4, move 3 stack on #4)
                      (7 moves     + 1 move +  7 moves) }
  Write ('Enter N: ');  Readln (N);
  A[1] := 1;
  for I := 2 to N do A[I] := A[I-1] * 2 + 1;
  Writeln (A[N])
end.
```

```
{3.10}
program Thr10T85;
{ -- This program will find sets of #s P, Q, R (P = Q x R). }
  var
    S, I, J, NU, X1, X2, Y1, Y2, Z2: Integer;
    X, C, Code: Integer;
    Dupl:       Boolean;
    Prod, Q, R: LongInt;
    P:          String[5];
    A:          Array [0..9] of Integer;

begin
  Write ('Enter S: ');  Readln (S);
  Q := S;
  repeat
    repeat
      Inc(Q);  X1 := Q div 10;  Y1 := Q mod 10;
    until X1 <> Y1;
    NU := 10000 div Q;
    for R := NU to 999 do begin
      Dupl := False;
      for I := 0 to 9 do A[I] := 0;
      X2 := R div 100;  C := R - X2 * 100;
      Y2 := C div 10;  Z2 := C - Y2 * 10;
      if (X2 <> Y2) and (Y2 <> Z2) and (X2 <> Z2) and
         (X1 <> X2) and (X1 <> Y2) and (X1 <> Z2) and
         (Y1 <> X2) and (Y1 <> Y2) and (Y1 <> Z2) then
      begin
        A[X1] := 1;  A[Y1] := 1;  A[X2] := 1;
        A[Y2] := 1;  A[Z2] := 1;
        Prod := Q * R;
        Str (Prod, P);
        if Length(P) = 5 then begin
          for I := 1 to 5 do begin
            Val(Copy(P, I, 1), X, Code);
            if A[X] = 1 then Dupl := True;
          end;
          for I := 1 to 4 do
            for J := I+1 to 5 do
              if Copy(P, I, 1) = Copy(P, J, 1) then Dupl := True;
          if not Dupl then begin
            Writeln ('P = ', P, '  Q = ', Q, '  R = ', R);
          end;
        end;  { -- if }
      end;  { -- if }
    end;  { -- for }
  until Q > 99;
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '86 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T86;
{ -- This program will print "THIS IS THE EASIEST PROGRAM!". }
uses Crt;

begin
  ClrScr;
  GotoXY (25, 12);  Writeln ('THIS IS THE EASIEST PROGRAM!');
end.


{1.2}
program One2T86;
{ -- This program will display the sum, difference, and product. }
  var
    Num1, Num2: Integer;

begin
  Write ('Enter two numbers: ');  Readln (Num1, Num2);
  Writeln ('SUM = ', Num1 + Num2);
  Writeln ('DIFFERENCE = ', Num1 - Num2);
  Writeln ('PRODUCT = ', Num1 * Num2);
end.


{1.3}
program One3T86;
{ -- This program will sum 1 + (1/2)^2 + (1/3)^3 + (1/4)^4 + ...
  -- until difference between it and the next term is within E. }
  var
    Sum, LastSum, E, Term, Prod: Real;
    I, J:                        Integer;

begin
  Write ('Enter test value E: ');  Readln (E);
  I := 1;
  Sum := 1;  LastSum := 0;
  while (Sum - LastSum) >= E do begin
    Inc(I);
    Term := 1.0 / I;  Prod := 1;
    for J := 1 to I do
      Prod := Prod * Term;
    LastSum := Sum;
    Sum := Sum + Prod;
  end;
  Writeln (LastSum :8:6);
end.
```

```
{1.4}
program One4T86;
{ -- This program will print a check given name and amount. }
uses Crt;
  var
    First, Last, Middle, Init, Amount: String[10];
    I:                                 Integer;

begin
  ClrScr;
  Write ('Enter first name: ');  Readln (First);
  Write ('Enter middle name: '); Readln (Middle);
  Write ('Enter last name: ');   Readln (Last);
  Init := Copy(Middle, 1, 1);
  Write ('Enter amount: ');      Readln (Amount);

  { -- Display border }
  GotoXY (1, 6);
  for I := 1 to 39 do
    Write ('*');
  for I := 1 to 9 do begin
    GotoXY (1, 6+I);  Write ('*');
    GotoXY (39, 6+I); Write ('*');
  end;
  GotoXY (1, 6+10);
  for I := 1 to 39 do
    Write ('*');

  GotoXY (3, 8);  Write ('BEN''S TOWING SERVICE');
  GotoXY (3, 9);  Write ('4563 WRECKER AVENUE');
  GotoXY (3, 10); Write ('WAVERLY, ARKANSAS 45632');
  GotoXY (4, 12); Write ('PAY TO THE ORDER OF ');
  Write (First, ' ', Init, '. ', Last);
  GotoXY (4, 14); Write ('THE SUM OF $', Amount);
  GotoXY (1, 22);
end.
```

```
{1.5}
program One5T86;
{ -- This program will determine which prisoners may be released.}
  var
    Cell: Array [1..100] of 0..1;
    I, J: Integer;

begin
  for I := 1 to 100 do
    Cell[I] := 1;  { -- Initialize all cells open }
  for I := 2 to 100 do begin
    J := 1;
    while J <= 100 do begin
      Cell[J] := 1 - Cell[J];
      Inc(J,I);
    end;
  end;

  for I := 1 to 100 do
    if Cell[I] = 1 then
      Writeln ('CELL ', I);
end.


{1.6}
program One6T86;
{ -- This program will determine how much money accumulates. }
  var
    Month, Deposit, Rate, Sum: Real;
    Year, J:                   Integer;

begin
  Write ('Enter monthly investment: ');
  Readln (Month);
  Write ('Enter end of year deposit: ');
  Readln (Deposit);
  Write ('Enter annual rate of interest: ');
  Readln (Rate);
  Writeln;
  Rate := Rate / (12*100);  { -- Rate per month in yr in percent }
  Sum := 0;
  for Year := 1 to 20 do begin
    for J := 1 to 12 do begin
      Sum := Sum + Month;
      Sum := Sum + Rate*Sum;
    end;
    Sum := Sum + Deposit;
  end;
  Writeln ('AMOUNT AT END OF YEAR 20 IS $', Sum: 4:2);
end.
```

```
{1.7}
program One7T86;
{ -- This program will drop g in words ending with ing or ings. }
  var
    I, L, LenWord: Integer;
    Sentence:      String[80];
    Word:          String[20];
    End1, End2:    String[4];
    Ch:            Char;

begin
  Write ('Enter sentence: ');  Readln (Sentence);
  Sentence := Sentence + ' ';
  L := Length(Sentence);
  I := 1;  Word := '';
  while I <= L do begin
    Ch := Sentence[I];
    if Ch <> ' ' then
      Word := Word + Ch
    else begin
      LenWord := Length(Word);
      if LenWord >= 4 then begin
        End1 := Copy(Word, LenWord-2, 3);
        End2 := Copy(Word, LenWord-3, 4);
        if End1 = 'ING' then
          Word := Copy(Word, 1, LenWord-1);
        if End2 = 'INGS' then
          Word := Copy(Word, 1, LenWord-2) + 'S';
      end;
      Write (Word, ' ');
      Word := '';
    end;
    Inc(I);
  end;
end.
```

```
{1.8}
program One8T86;
{ -- This program simulates the population growth of rabbits. }
  var
    Init, OverPop: Integer;
    Month, I:      Integer;
    Pop:           Real;
    Dieing:        Boolean;

begin
  Write ('Enter initial population: ');  Readln (Init);
  Write ('Enter point of over population: ');  Readln (OverPop);
  Writeln;
  Pop := Init;
  Dieing := (Pop >= OverPop);
  for Month := 1 to 23 do begin
    If Dieing then
      If (Pop < 2/3 * Init) then
        begin
          Dieing := False;
          Pop := Pop + Pop * 0.2;
        end
      else
        Pop := Pop - Pop * 0.15
    else
      if (Pop >= OverPop) then
        begin
          Dieing := True;
          Init := Trunc(Pop);
          Pop := Pop - Pop * 0.15;
        end
      else
        Pop := Pop + Pop * 0.2;

    Writeln ('POPULATION FOR MONTH ', Month, ' IS ', Pop :2:0);
  end;
end.
```

```
{1.9}
program One9T86;
{ -- This program doubles every e that appears as a single e. }
  var
    Sentence:           String[200];
    LastCh, Ch, NextCh: Char;
    I:                  Integer;
begin
  Write ('Enter sentence: ');  Readln (Sentence);
  I := 1;  LastCh := ' ';
  repeat
    Ch := Sentence[I];
    NextCh := Sentence[I+1];
    if (Ch = 'E') and (LastCh <> 'E') and (NextCh <> 'E') then
      Write ('E');
    Write (Ch);
    Inc(I);
    LastCh := Ch;
  until I = Length(Sentence);
  if (NextCh = 'E') and (LastCh <> 'E') then
    Write ('E');
  Write (NextCh);
end.


{1.10}
program One10T86;
{ -- This program will display common elements of two lists. }
  var
    I, J:    Integer;
    A, B, C: Array [1..12] of Integer;
begin
  for I := 1 to 12 do begin
    Write ('Enter ', I, ' of 12: ');  Readln (A[I]);
  end;
  Writeln;
  for I := 1 to 11 do begin
    Write ('Enter ', I, ' of 11: ');  Readln (B[I]);
  end;

  for I := 1 to 12 do C[I] := 0;
  for I := 1 to 12 do
    for J := 1 to 11 do
      if A[I] = B[J] then C[I] := 1;

  for I := 1 to 12 do
    for J := I + 1 to 12 do
      if (A[I] = A[J]) and (C[J] > 0) then
        Inc(C[J]);

  for I := 1 to 12 do
    if C[I] = 1 then
      Write (A[I], '  ');
end.
```

```
{2.1}
program Two1T86;
{ -- This program will right justify sentence within 65 columns. }
  const
    Col: Integer = 65;
  var
    Sentence, Just:  String[65];
    Word:            Array [1..20] of String[12];
    Ch:              Char;
    I, L, Extra, Ex: Integer;
    WordNum:         Integer;
    TotalCh, SpAve:  Integer;

begin
  Write ('Enter Sentence: ');  Readln (Sentence);
  Sentence := Sentence + ' ';
  L := Length(Sentence);
  I := 1;  WordNum := 1;  Word[WordNum] := '';
  TotalCh := 0;
  { -- Parse Words and calculate Total # of Characters in words }
  while (I <= L) do begin
    Ch := Sentence[I];
    if Ch <> ' ' then
      Word[WordNum] := Word[WordNum] + Ch
    else
      if Word[WordNum] > '' then begin
        TotalCh := TotalCh + Length(Word[WordNum]);
        Inc(WordNum);
        Word[WordNum] := '';
      end;
    Inc(I);
  end;
  Dec(WordNum);

  { -- Display words with SpAve spaces between each one. }
  SpAve := (Col - TotalCh) div (WordNum - 1);
  Extra := (Col - TotalCh) - (SpAve * (WordNum-1));
  for I := 1 to WordNum do begin
    If I <= Extra then Ex := 1
      else Ex := 0;
    Write (Word[I], ' ': SpAve + Ex);
  end;
end.
```

```
{2.2}
program Two2T86;
{ -- This program will produce a repeating pattern with XXX -- }
  var
    X1, X2, D1, D2: String[7];
    TotalXD, Row:   Integer;
    NumX, Rows, I:  Integer;

begin
  Write ('Enter total number of X''s and -''s: ');
  Readln (TotalXD);
  Write ('Enter number of X''s: '); Readln (NumX);
  Write ('Enter number of rows: '); Readln (Rows);

  X1 := '';  X2 := '';  D1 := '';  D2 := '';
  for I := 1 to NumX do begin
    X1 := X1 + 'X';
    D2 := D2 + '-';
  end;
  for I := 1 to TotalXD - NumX do begin
    X2 := X2 + 'X';
    D1 := D1 + '-';
  end;

  for Row := 1 to Rows do begin
    if Row mod 2 = 1 then
      for I := 1 to 4 do
        Write (X1, D1)
    else
      for I := 1 to 4 do
        Write (D2, X2);
    Writeln;
  end;
end.
```

```
{2.3}
program Two3T86;
{ -- This program will code or decode a message. }
  var
    Option, I: Integer;
    St1, St2:  String[27];
    Message:   String[80];
    Ch:        Char;

begin
  St1 := 'ZXCVBNMASDFGHJKLQWERTYUIOP ';
  St2 := 'ABCDEFGHIJKLMNOPQRSTUVWXYZ ';
  repeat
    Writeln;
    Writeln ('1) ENCODE');
    Writeln ('2) DECODE');
    Writeln ('3) END');
    Write   ('Choose: ');  Readln (Option);
    if Option < 3 then begin
      Write ('Enter message: ');  Readln (Message);
      for I := 1 to Length(Message) do begin
        Ch := Message[I];
        if Ch <> ' ' then
          if Option = 1 then  { -- Code message }
            Ch := St1[Ord(Ch) - 64]
          else                { -- Decode message }
            Ch := St2[Pos(Ch, St1)];
        Write (Ch);
      end;
      Writeln;
    end;
  until Option = 3;
end.
```

```
{2.4}
program Two4T86;
{ -- This program finds the unique mode of a set of 15 numbers. }
  var
    A, C:          Array [1..15] of Integer;
    I, J, K, Max:  Integer;
    Mode:          Integer;
    ModeExist:     Boolean;

begin
  for I := 1 to 15 do begin
    Write ('Enter number ', I, ': ');  Readln (A[I]);
  end;

  Max := 1;
  for I := 1 to 14 do begin
    C[I] := 1;
    for J := I + 1 to 15 do
      if A[I] = A[J] then begin
        Inc(C[I]);  { -- Has # of duplicates of elements }
        if C[I] > Max then
          Max := C[I];
      end;
  end;

  { -- Mode exists if only one element occurs Max # of times. }
  ModeExist := False;
  for I := 1 to 14 do
    if (C[I] = Max) then
      if not ModeExist then
        begin
          Mode := A[I];  ModeExist := True;
        end
      else begin
        Writeln ('NO UNIQUE MODE');  Exit;  end;

  if ModeExist then
    Writeln ('MODE IS ', Mode)
  else
    Writeln ('NO UNIQUE MODE');
end.
```

```
{2.5}
program Two5T86;
{ -- This program simulates transactions to a savings accounts. }
  const
    Rate: Real = 0.07;
  var
    Option:                                  Integer;
    Balance, Deposit, Withdrawal, Credit: Real;

begin
  Write ('Enter original balance: ');  Readln (Balance);
  Writeln;
  repeat
    Writeln ('1. MAKE A DEPOSIT');
    Writeln ('2. MAKE A WITHDRAWAL');
    Writeln ('3. CREDIT INTEREST');
    Writeln ('4. END');
    Write ('Enter option: ');  Readln (Option);  Writeln;
    case Option of
      1: begin
           Write ('Enter amount to deposit: ');  Readln (Deposit);
           Writeln ('BALANCE BEFORE TRANSACTION $', Balance: 7:2);
           Balance := Balance + Deposit;
           Writeln ('MAKE A DEPOSIT');
         end;
      2: begin
           Write ('Enter amount to withdraw: ');
           Readln (Withdrawal);
           Writeln ('BALANCE BEFORE TRANSACTION $', Balance: 7:2);
           Balance := Balance - Withdrawal;
           Writeln ('MAKE A WITHDRAWAL');
         end;
      3: begin
           Writeln ('BALANCE BEFORE TRANSACTION $', Balance: 7:2);
           Credit := Balance * Rate/12;
           Writeln ('CREDIT INTEREST OF $', Credit: 4:2);
           Balance := Balance + Credit;
         end;
    end;

    if Option < 4 then Write ('NEW ')
      else Write ('FINAL ');
    Writeln ('BALANCE $', Balance: 7:2);
    Writeln;
  until Option = 4;
end.
```

```
{2.6}
program Two6T86;
{ -- This program will sum two positive big numbers. }
  var
    St1, St2:    String[38];
    A, B, C:     Array [1..39] of Integer;
    I, L1, L2,
    MaxL, Carry: Integer;
    Ch:          Char;

begin
  Write ('Enter first number:  ');  Readln (St1);
  Write ('Enter second number: ');  Readln (St2);
  for I := 1 to 39 do begin
    A[I] := 0;  B[I] := 0;
  end;
  L1 := Length(St1);  L2 := Length(St2);
  { -- Put 1st number in A[1..L1], 2nd number in B[1..L2] }
  for I := 1 to L1 do begin
    Ch := St1[ L1-I+1 ];
    A[I] := Ord(Ch) - Ord('0');
  end;
  for I := 1 to L2 do begin
    Ch := St2[ L2-I+1 ];
    B[I] := Ord(Ch) - Ord('0');
  end;

  if L1 > L2 then MaxL := L1
    else MaxL := L2;
  Carry := 0;
  { -- Calculate sum in C[1..MaxL] }
  for I := 1 to MaxL do begin
    C[I] := A[I] + B[I] + Carry;
    if C[I] > 9 then begin
      C[I] := C[I] - 10;
      Carry := 1;
    end
    else Carry := 0;
  end;
  if Carry = 1 then begin
    MaxL := MaxL + 1;
    C[MaxL] := 1;
  end;

  Write ('SUM IS ');
  for I := MaxL downto 1 do
    Write (C[I]);
end.
```

```
{2.7}
program Two7T86;
{ -- This program will perform conversions. }
  const
    Dec: Array [1..6] of String[11] =
      ('INCHES', 'FEET', 'MILES', 'OUNCES', 'POUNDS', 'GALLONS');
    Con: Array [1..6] of Real =
      (2.54, 0.3048, 1.6093, 28.35, 0.4536, 3.7854);
    Met: Array [1..6] of String[11] =
      ('CENTIMETERS', 'METERS', 'KILOMETERS', 'GRAMS',
       'KILOGRAMS', 'LITERS');
  var
    Option, I: Integer;
    X, Y:      Real;
    St:        String[30];

begin
  repeat
    Writeln;
    { -- Display menu options }
    for I := 1 to 6 do begin
      Write (I: 2, ' ');
      if I mod 2 = 1 then
        begin
          St := Met[(I+1) div 2] + ' TO ' + Dec[(I+1) div 2];
          Write (St, ' ': 23 - Length(St));
          Write (I+6: 2, ' ');
          St := Met[(I+7) div 2] + ' TO ' + Dec[(I+7) div 2];
        end
      else
        begin
          St := Dec[I div 2] + ' TO ' + Met[I div 2];
          Write (St, ' ': 23 - Length(St));
          Write (I+6: 2, ' ');
          St := Dec[(I+6) div 2] + ' TO ' + Met[(I+6) div 2];
        end;
      Writeln (St);
    end;
    Writeln ('13 END' :32);
    Write ('Enter option: ');  Readln (Option);

    if Option < 13 then
    if Option mod 2 = 1 then  { -- Convert Metric to English }
      begin
        Write ('Enter number of ', Met[(Option + 1) div 2],': ');
        Readln (X);
        Y := X / Con[(Option + 1) div 2];
        Write ('THIS IS EQUIVALENT TO ', Y:7:3, ' ');
        Writeln (Dec[(Option+1) div 2]);
      end
    else  { -- Convert English to Metric }
      begin
        Write ('Enter number of ', Dec[Option div 2], ': ');
        Readln (X);
        Y := X * Con[Option div 2];
```

```
        Write ('THIS IS EQUIVALENT TO ', Y:7:3, ' ');
        Writeln (Met[Option div 2]);
      end;
  until Option = 13;
end.


{2.8}
program Two8T86;
{ -- This program will generate a mortgate amortization. }
uses Crt;
  var
    Rate, Principal, Payment: Real;
    Years, I, C, Month:       Integer;
    YI, TI, MI, MP, OldP:     Real;
    Ch: Char;

function Power({using} X: Real; {raised to the} Y: Integer):
                                                  {giving} Real;
{ -- This function simulates the ^ (power) symbol (X to the Y) }
  var
    I: Integer;
    P: Real;
begin
  P := X;
  for I := 1 to Y-1 do
    P := P * X;
  Power := P;
end;

begin
  Write ('Enter principal: ');           Readln (Principal);
  Write ('Enter % rate of interest: ');  Readln (Rate);
  Write ('Enter term in years: ');       Readln (Years);
  Write ('Enter # of month in year for first payment: ');
  Readln (Month);

  Rate := Rate / (12 * 100);
  Payment := (Rate * Power((1+Rate),(Years*12)))/
             (Power((1+Rate),(12*Years)) -1) * Principal;
  C := Month - 1;  OldP := Principal;
  Rate := Rate * 12;  YI := 0;  TI := 0;
  Writeln ('INTEREST         PRINCIPAL');

  for I := 1 to Years*12 do begin
    MI := OldP * Rate/12;
    MP := Payment - MI;
    OldP := OldP - MP;
    Writeln ('$', MI: 6:2, ' ':10, '$', OldP :8:2);
    C := C + 1;  YI := YI + MI;
    if C mod 12 = 0 then begin
      Writeln;
      Writeln ('YEAR''S INTEREST', '  $', YI: 8:2);
      TI := TI + YI;  YI := 0;
      Ch := ReadKey;
```

```
    end;
  end;

  if Month <> 1 then begin
    Writeln;
    Writeln ('YEAR''S INTEREST', '  $', YI: 8:2);
    TI := TI + YI;
    Ch := ReadKey;
  end;
  Writeln ('TOTAL INTEREST   $', TI: 8:2);
  Writeln ('MONTHLY PAYMENT  $', Payment: 8:2);
end.


{2.9}
program Two9T86;
{ -- This program calculates the value of sine(x) by a series. }
  var
    N, X, Sum, Factorial, Term: Real;
    I, J, Power:                Integer;

begin
  Write ('Enter N degrees: ');  Readln (N);
  Sum := 0;
  if N > 180 then
    X := Pi * ((360-N)/180)
  else
    X := Pi * (N/180);
  Power := -1;
  for I := 1 to 6 do begin
    Power := Power + 2;
    Factorial := 1;
    for J := 1 to Power do
      Factorial := Factorial * J;
    Term := 1;
    for J := 1 to Power do
      Term := Term * X;
    Term := Term / Factorial;
    if I mod 2 = 1 then
      Sum := Sum + Term
    else
      Sum := Sum - Term;
  end;

  if N > 180 then begin
    Sum := -1 * Sum;  X := Pi * (N/180);
  end;
  Writeln ('PARTIAL SUM = ', Sum :9:7);
  Writeln ('ACTUAL SINE = ', Sin(X) :8:7);
end.
```

```
{2.10}
program Two10T86;
{ -- This program will convert a Roman Numeral to Arabic form. }
  const
    RN: String[7] = 'MDCLXVI';
    RV: Array [1..7] of Integer = (1000, 500, 100, 50, 10, 5, 1);
  var
    RomNum:        String[12];
    I, Ind1, Ind2: Integer;
    L, Arabic:     Integer;
    Ch, NextCh:    Char;

begin
  Write ('Enter Roman Numeral: ');  Readln (RomNum);
  L := Length (RomNum);  I := 1;  Arabic := 0;
  while (I < L) do begin
    Ch     := RomNum[I];    Ind1 := Pos(Ch, RN);
    NextCh := RomNum[I+1];  Ind2 := Pos(NextCh, RN);
    if Ind1 <= Ind2 then { -- value of first is greater or equal}
      Arabic := Arabic + RV[Ind1]
    else begin  { -- value of first is less than second }
      Arabic := Arabic + RV[Ind2] - RV[Ind1];
      Inc(I);
    end;
    Inc(I);
  end;

  if I = L then begin  { -- Last numeral was not done }
    Ch := RomNum[I];  Ind1 := Pos(Ch, RN);
    Arabic := Arabic + RV[Ind1];
  end;
  Writeln ('ARABIC = ', Arabic);
end.
```

```
{3.1}
program Thr1T86;
{ -- This program produces monthly calendars for the year 1986. }
uses Crt;
  const
    Mo: Array[1..12] of String[9] = ('JANUARY','FEBRUARY',
      'MARCH','APRIL','MAY','JUNE','JULY','AUGUST','SEPTEMBER',
      'OCTOBER','NOVEMBER','DECEMBER');
    Days: Array[1..12] of Integer =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    D: Array[1..7] of Char = ('S', 'M', 'T', 'W', 'T', 'F', 'S');

  var
    I, M, Col, Day: Integer;
    Ch: Char;

begin
  ClrScr;
  Writeln (' ':14, '1986');  Writeln;
  for M := 1 to 12 do begin
    { -- Display Month name and Day initials. }
    if M > 1 then ClrScr;
    Writeln (' ':12, Mo[M]);  Writeln;
    for I := 1 to 7 do
      Write (D[I]: 4);
    Writeln;

    { -- Display Day numbers in proper column. }
    if M = 1 then Col := 4;
    if Col > 1 then
      Write (' ': (Col-1)*4);
    for Day := 1 to Days[M] do begin
      Write (Day: 4);
      if Col < 7 then
        Col := Col + 1
      else begin
        Col := 1;  Writeln;
      end;
    end;
    Ch := ReadKey;
  end;
end.
```

```
{3.2}
program Thr2T86;
{ -- This program finds the root of a 5th degree polynomial }
{ -- of the form Ax^5 + Bx^4 + Cx^3 + Dx^2 + Ex + F = 0.    }
  var
    A, B, C, D, E, F: Real;
    X, X1, X2:  Real;

function Y(X,A,B,C,D,E,F: Real):Real;
{ -- This function returns value of Y given coefficients and X. }
begin
  Y := A*X*X*X*X*X + B*X*X*X*X + C*X*X*X + D*X*X + E*X + F;
end;

begin
  Write ('Enter coefficients A,B,C,D,E,F: ');
  Readln (A,B,C,D,E,F);
  { -- This algorithm finds 1 and only 1 root (closest to x=0) }
  X1 := -1.0;  X2 := 1.0;
  { -- Find sign change between X1 and X2. }
  while Y(X1,A,B,C,D,E,F) * Y(X2,A,B,C,D,E,F) > 0 do begin
    X1 := X1 - 1;  X2 := X2 + 1;
  end;
  { -- Use binary search to find root. }
  while X2 - X1 > 0.000005 do begin
    X := (X1 + X2) / 2;
    if Y(X,A,B,C,D,E,F) * Y(X1,A,B,C,D,E,F) > 0 then X1 := X
      else X2 := X;
  end;
  Writeln ('ROOT = ', X: 7:5);
end.
```

```
{3.3}
program Thr3T86;
{ -- This program changes a number from one base to another. }
  const
    D: String[36] = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
  var
    A, B, I, J, Ex, X: Integer;
    N, Pow:            Real;
    NumSt:             String[10];

begin
  Write ('Enter base A: ');   Readln (A);
  Write ('Enter base B: ');   Readln (B);
  Write ('Enter original number: '); Readln (NumSt);
  Writeln;  Write (NumSt, ' BASE ', A, ' EQUALS ');
  { -- Convert Num to Base 10 from base A. }
  N := 0;
  for I := 1 to Length(NumSt) do begin
    Pow := 1;
    for J := 1 to Length(NumSt)-I do Pow := Pow * A;
    N := N + (Pos(Copy(NumSt,I,1),D) - 1) * Pow;
  end;

  Ex := 0;  Pow := 1;
  while Pow <= N do begin
    Inc(Ex);  Pow := Pow * B;
  end;
  Dec(Ex);

  { -- Convert Num to Base B from Base 10. }
  for I := Ex downto 0 do begin
    Pow := Pow / B;
    X := Trunc(N / Pow + 0.01);
    Write (D[X+1]);
    N := N - X*Pow;
  end;
  Write (' BASE ', B);
end.
```

```
{3.4}
program Thr4T86;
{ -- This program will update customers account by SSN's. }
  var
    SS:    Array[1..6] of String[9];
    N:     Array[1..6] of String[12];
    A:     Array[1..6] of String[41];
    B:     Array[1..6] of Real;
    SSN:   String[10];
    Temp:  String[41];
    I,J,L: Integer;
    Ch:    Char;
    Trans: Real;
    P1,P2: Integer;

begin
  SS[1] := '234567890'; N[1] := 'JOHN SMITH  ';
  SS[2] := '564783219'; N[2] := 'GAIL HUSTON ';
  SS[3] := '873421765'; N[3] := 'TIM JONES   ';
  SS[4] := '543876543'; N[4] := 'JILL RUPERTS';
  SS[5] := '345212342'; N[5] := 'AL BROWN    ';
  SS[6] := '565656565'; N[6] := 'KERMIT TEU  ';
  A[1]  := '1234 ANYWHERE LANE, EXIST, KANSAS 66754  ';
  A[2]  := '543 SOUTH THIRD, BIG TOWN, TEXAS 88642   ';
  A[3]  := '2387 PALM PLACE, NOME, ALASKA 77643      ';
  A[4]  := '4536 123RD STREET, TINY TOWN, MAINE 76765';
  A[5]  := 'PO BOX 234, TINSEL TOWN, CALIFORNIA 77654';
  A[6]  := '1234 LOST LANE, WIMPLE, WISCONSIN 66543  ';
  B[1]  :=  345.78;
  B[2]  := 2365.89;
  B[3]  := 6754.76;
  B[4]  :=   45.18;
  B[5]  := 3456.09;
  B[6]  :=   78.36;

  Write ('Enter SSN: ');  Readln (SSN);
  while SSN <> '000000000' do begin
    I := 1;
    while (SS[I] <> SSN) and (I < 6) do I := I + 1;
    Write ('Enter C for Charge or P for Payment: ');  Readln(Ch);
    Write ('Enter amount of transaction: ');  Readln(Trans);
    if Ch = 'C' then
      B[I] := B[I] - Trans
    else
      B[I] := B[I] + Trans;
    Writeln;
    Writeln ('NEW BALANCE IS $', B[I]: 5:2);
    Writeln;
    Write ('Enter SSN: ');  Readln (SSN);
  end;
  { -- Sort customers in decreasing order according to balance. }
  for I := 1 to 5 do
    for J := I + 1 to 6 do
      if B[I] < B[J] then begin
        Temp := SS[I]; SS[I] := SS[J];  SS[J] := Temp;
```

```
        Temp := N[I];   N[I] := N[J];    N[J] := Temp;
        Temp := A[I];   A[I] := A[J];    A[J] := Temp;
        Trans := B[I];  B[I] := B[J];    B[J] := Trans;
      end;
  { -- Display report }
  Writeln;
  Write   ('SSN', ' ':8, 'NAME', ' ': 10, 'ADDRESS', ' ':2);
  Writeln ('BALANCE': 18);  Writeln;
  for I := 1 to 6 do begin
    Temp := SS[I] + '  ' + N[I] + '  ';
    Write (Temp);
    L := Length(Temp) - 1;
    P1 := Pos(',', A[I]);  Delete(A[I], P1, 1);
    P2 := Pos(',', A[I]);
    Write (Copy(A[I], 1, P1 - 1));
    Writeln ('$': 22 - P1, B[I]:7:2);
    Writeln (' ': L, Copy(A[I], P1, P2 - P1));
    Writeln (' ': L, Copy(A[I], P2+1, Length(A[I]) - P2 - 1));
  end;
  Writeln;
end.
```

```
{3.5}
program Thr5T86;
{ -- This program will print the product of 2 large decimals. }
  var
    AStr, BStr:                        String[31];
    LenA, LenB, ADec, BDec, RDigits: Integer;
    A, B, Prod:                        Array[1..61] of Integer;
    I, J, S, Carry, Base:              Integer;
    Sign: -1..1;

begin
  Write ('Enter first number: ');   Readln (AStr);
  Write ('Enter second number: ');  Readln (BStr);

  { -- Determine # of Digits to the right of decimal in product }
  ADec := Pos ('.', AStr);  BDec := Pos ('.', BStr);
  Delete (AStr, ADec, 1);   Delete (BStr, BDec, 1);
  LenA := Length(AStr);     LenB := Length(BStr);
  RDigits := LenA - ADec + LenB - BDec + 2;

  { -- Store String digits into numerical arrays. }
  for I := LenA downto 1 do
    A[LenA-I+1] := Ord(AStr[I]) - 48;
  for I := LenB downto 1 do
    B[LenB-I+1] := Ord(BStr[I]) - 48;
  for I := 1 to 61 do Prod[I] := 0;

  { -- Multiply 2 numbers as a person would with carries. }
  for I := 1 to LenB do begin
    Carry := 0;
    for J := 1 to LenA do begin
      S := I + J - 1;
      Prod[S] := Prod[S] + B[I]*A[J] + Carry;
      Carry := Prod[S] div 10;
      Prod[S] := Prod[S] - Carry*10;
    end;
    If Carry > 0 then Prod[S+1] := Carry;
  end;

  { -- Display digits of product before decimal }
  Write ('PRODUCT = ');
  if Carry > 0 then Inc(S);
  if S > RDigits then
    for I := S downto RDigits+1 do
      Write (Prod[I])
   else
     Write ('0');
  Write ('.');
  { -- Display digits after decimal. }
  for I := RDigits downto 1 do
    Write (Prod[I]);
end.
```

```
{3.6}
program Thr6T86;
{ -- This program will determine if a # can become palindrome. }
  var
    B, Rev:           Array[1..50] of Integer;
    I, L, Try, Carry: Integer;
    Pal:              Boolean;
    NumSt:            String[10];

begin
  Write ('Enter number: ');  Readln (NumSt);
  L := Length(NumSt);
  for I := 1 to L do
    B[L-I+1] := Ord(NumSt[I]) - 48;
  Try := 0;

  repeat
    { -- Test for Palindrome }
    Pal := True;
    for I := 1 to (L div 2) do
      if B[I] <> B[L-I+1] then Pal := False;

    { -- Add reverse of number to itself. }
    if not Pal then begin
      for I := 1 to L do Rev[I] := B[L-I+1];
      Carry := 0;
      for I := 1 to L do begin
        B[I] := B[I] + Rev[I] + Carry;
        Carry := B[I] div 10;
        B[I] := B[I] - Carry*10;
      end;
      if Carry = 1 then begin
        Inc(L);  B[L] := 1;
      end;
      Inc(Try);
    end;
  until Pal or (Try > 23);

  { -- Display # if Palindrome else say it is not. }
  if Pal then begin
    for I := L downto 1 do Write (B[I]);
    Writeln (' IS A PALINDROME');
    end
  else
    Writeln ('CANNOT GENERATE A PALINDROME');
end.
```

```
{3.7}
program Thr7T86;
{ -- This program will solve an N x N system of equations. }
  var
    C:              Array[1..5,1..6] of Real;
    N, Row, Col, R: Integer;
    Den, X:         Real;

begin
  { -- Enter values in C array }
  Write ('Enter N: ');  Readln (N);
  for Row := 1 to N do begin
    Writeln ('Enter coefficients for Row ', Row);
    for Col := 1 to N do begin
      Write ('Co', Col, ': ');
      Readln (C[Row,Col]);
    end;
    Write ('Enter constant: ');  Readln (C[Row, N+1]);
  end;

  { -- Make main diagonals all 1s with 0s to the left. }
  for Row := 1 to N do begin
    Den := C[Row, Row];
    for Col := Row to N+1 do
      C[Row, Col] := C[Row, Col] / Den;
    for R := Row+1 to N do begin
      X := C[R, Row];
      for Col := Row to N+1 do
        C[R,Col] := C[R,Col] - X * C[Row,Col];
    end;
  end;

  { -- Make 0s on right of 1s on main diagonal, (not constants).}
  for Row := N downto 1 do
    for R := Row-1 downto 1 do begin
      X := C[R, Row];
      for Col := Row to N+1 do
        C[R,Col] := C[R,Col] - X * C[Row,Col];
    end;

  { -- Display solution }
  Write ('(', C[1,N+1]: 1:0);
  for Row := 2 to N do
    Write (', ', C[Row,N+1]: 1:0);
  Writeln (')');
end.
```

```
{3.8}
program Thr8T86;
{ -- This program prints Kth, 2*Kth, and 3*Kth permutations. }
  var
    F, I, J, K,
    L, KK, T, X, S: Integer;
    AStr:           String[7];
    A:              Array[1..7] of Char;
    B:              Array[1..7] of 0..1;
    Temp:           Char;
    Fact:           Array[1..7] of Integer;
    Quit:           Boolean;
begin
  Write ('Enter word: ');  Readln (AStr);
  Write ('Enter K: ');     Readln (K);
  L := Length (AStr);
  { -- Store and alphabetize letters. }
  for I := 1 to L do A[I] := AStr[I];
  for I := 1 to L-1 do
    for J := I+1 to L do
      if A[I] > A[J] then begin
        Temp := A[I];  A[I] := A[J];  A[J] := Temp;
      end;

  { -- Compute Factorials F[3] = 2!, F[4] = 3!... }
  for I := 1 to L do begin
    F := 1;
    for J := 1 to I-1 do F := F * J;
    Fact[I] := F;
  end;

  { -- Generate permutations in order. }
  for T := 1 to 3 do begin
    KK := K*T-1;
    for I := 1 to 7 do B[I] := 0;
    for I := L downto 1 do begin
      X := KK div Fact[I];  S := 0;
      J := 1;  Quit := False;
      repeat
        if B[J] = 0 then begin
          Inc(S);
          if S > X then begin
            B[J] := 1;
            Write (A[J]);
            Quit := True;
          end;
        end;
        Inc(J);
      until (J > L) or Quit;
      KK := KK - Fact[I]*X;
    end;  { -- for I }
    Write('  ');
  end;  { -- for T }
end.
```

```
{3.9}
program Thr9T86;
{ -- This program will solve cryptarithm puzzle ABB - CB = DEF. }
{ -- F = 0 since B-B=0.  A=D+1 or A=D since CB is 2 digits,
     but A<>D. D>B, otherwise D=A. Since B<C, B<9, => E=10+B-C. }
  var
    A, B, C, D, E, F, Tot: Integer;

begin
  Tot := 0;
  for B := 1 to 8 do
    for C := B+1 to 9 do
      for D := 1 to 8 do begin
        F := 0;
        A := D + 1;
        E := 10 + B - C;
        if not ((A=B) or (A=C) or (A=D) or (A=E) or (A=F) or
                (B=C) or (B=D) or (B=E) or (B=F) or (C=D) or
                (C=E) or (C=F) or (D=E) or (D=F)) then begin
           Tot := Tot + 1;
           Writeln (A,B,B,' - ',C,B,' = ',D,E,F,'  NUMBER ',Tot);
         end;
      end;  { -- for D }
  Writeln;
  Writeln ('TOTAL NUMBER OF SOLUTIONS = ',Tot);
end.
```

```
{3.10}
program Thr10T86;
{ -- This program will find all 2-digit integers equal to the sum
     of integers in which each digit 0-9 is used exactly once. }
{ -- Array D is array of digits to appear in Ten's position.
  -- C is count of how many digits are in array D.
  -- S is sum of digits not in array D
  -- F is flag array showing which digits are not in array D. }

  var
    I, J, K, C, DD, N, S, D1, D2, D3, P: Integer;
    F, D: Array[0..9] of Integer;

procedure CheckCondition;
{ -- This procedure will Check the condition. }
begin
  S := 0;  F[0] := 1;
  for I := 1 to 9 do F[I] := 0;
  for I := 1 to 9 do
    if not ((C=1) and (I=D1) or (C=2) and ((I=D1) or (I=D2)) or
            (C=3) and ((I=D1) or (I=D2) or (I=D3))) then begin
      S := S + I;  F[I] := 1;
    end;
  if C = 1 then DD := D1;
  if C = 2 then DD := D1 + D2;
  if C = 3 then DD := D1 + D2 + D3;
  if DD * 10 + S = N then begin
    Write (N, ' = ');
    K := 0;
    for J := 1 to C do begin
      while F[K] = 0 do K := K + 1;
      Write (D[J], K, ' + ');
      Inc(K);
    end;
    for I := K to 9 do begin
      if F[I] = 1 then begin
        Write (I);
        if I < 9 then Write (' + ');
      end;
    end;
    Writeln;
    P := 1;
  end;
end;

begin
  for N := 45 to 99 do begin
    for D1 := 1 to 2 do begin
      D[1] := D1;
      for D2 := D1+1 to 3 do begin
        D[2] := D2;
        for D3 := D2+1 to 4 do begin
          D[3] := D3;  C := 3;  CheckCondition;
        end;
      end;
```

```
    end;  { -- for D1}
    D3 := 0;
    if P <> 1 then begin
      for D1 := 1 to 2 do begin
        D[1] := D1;
        for D2 := D1+1 to 3 do begin
          D[2] := D2;  C := 2;  CheckCondition;
        end;
      end;
      D2 := 0;
      if P <> 1 then begin
        for D1 := 1 to 6 do begin
          D[1] := D1;  C := 1;  CheckCondition;
        end;
        if N = 45 then begin
          Write (N, ' = ');
          K := 0;
          for I := K to 9 do begin
            if F[I] = 1 then begin
              Write (I);
              if I < 9 then Write (' + ');
            end;
          end;
          Writeln;
          P := 1;
        end;
      end;  { -- if P<>1 }
    end;  { -- if P<>1 }
    P := 0;
  end;  { -- for N }
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '87 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T87;
{ -- This program will print out the sign of a given number. }
  var
    Num: Real;

begin
  Write ('Enter number: ');
  Readln (Num);
  if Num > 0 then
    Writeln ('POSITIVE')
  else if Num < 0 then
    Writeln ('NEGATIVE')
  else
    Writeln ('ZERO');
end.

{1.2}
program One2T87;
{ -- This program will sum the numbers n, n+1, ... n+20. }
  var
    N, I, Sum: Integer;

begin
  Write ('Enter n: ');  Readln (N);
  Sum := 0;
  for I := 0 to 20 do
    Sum := Sum + N + I;
  Writeln ('SUM = ', Sum);
end.

{1.3}
program One3T87;
{ -- This program will print PROBLEM THREE diagonally. }
uses Crt;
  const
    St = 'PROBLEM THREE';
  var
    Row, Col, I, L: Byte;

begin
  ClrScr;
  L := Length (St);
  Row := (24 - L) div 2;
  Col := (80 - L) div 2;
  for I := 1 to L do begin
    GotoXY (Col+I, Row+I);
    Write (Copy (St,I,1));
  end;
end.
```

```
{1.4}
program One4T87;
{ -- This program displays the numbers on the sides of a die. }
  var
    Top, Front, Right: Byte;

begin
  Write ('Enter number on top: ');    Readln (Top);
  Write ('Enter number on front: ');  Readln (Front);
  Write ('Enter number on right: ');  Readln (Right);

  Writeln ('TOP= ',    Top);
  Writeln ('FRONT= ',  Front);
  Writeln ('RIGHT= ',  Right);
  Writeln ('BOTTOM= ', 7 - Top);
  Writeln ('BACK= ',   7 - Front);
  Writeln ('LEFT= ',   7 - Right);
end.


{1.5}
program One5T87;
{ -- This program will fill the screen with random characters. }
uses Crt;
  var
    Row, Col: Byte;

begin
  Randomize;
  for Row := 1 to 24 do
    for Col := 1 to 80 do
      Write( Chr (Random (96) + 33) );
  repeat until KeyPressed;
  ClrScr;
end.


{1.6}
program One6T87;
{ -- This program will display a rectangular array of periods. }
uses Crt;
  var
    Row1, Col1, Row2, Col2, I, J: Byte;

begin
  Write ('Enter coordinates: ');
  Readln (Row1, Col1, Row2, Col2);
  ClrScr;
  for I := Row1 to Row2 do
    for J := Col1 to Col2 do begin
      GotoXY (J, I);  Write ('.');
    end;
end.
```

```
{1.7}
program One7T87;
{ -- This program will generate 10 random numbers given a seed. }
  var
    Seed, I: Integer;

begin
  Write ('Enter seed: ');  Readln (Seed);
  for I := 1 to 10 do begin
    Seed := (Seed * 421 + 1) mod 100;
    Writeln (Seed);
  end;
end.


{1.8}
program One8T87;
{ -- This program will determine the mass of a fish tank. }
  var
    K, L, W, H, Mass, InchCubed: Real;

begin
  Write ('Enter K, L, W, H: ');  Readln (K, L, W, H);
  InchCubed := 2.54 * 2.54 * 2.54;
  Mass := L * 12 * W * 12 * H * 12 * InchCubed;
  Mass := Mass / 1000 + K;
  Writeln (Mass: 8:2, ' KILOGRAMS');
end.


{1.9}
program One9T87;
{ -- This program will display 21 rows of letters. }
uses Crt;
  var
    Row, I: Integer;
    Ch:     Char;

begin
  ClrScr;
  for Row := 1 to 21 do begin
    Ch := Chr(64 + Row);
    if Row mod 2 = 1 then
      for I := 1 to 31 do
        Write (Ch)
    else begin
      Write (Ch);
      for I := 1 to 10 do
        Write ('  ', Ch);
    end;
    Writeln;
  end;
end.
```

```
{1.10}
program One10T87;
{ -- This program will display the time needed to read a book. }
  const
    Title : Array [1..4] of String[30] =
      ('THE HISTORY OF THE COMPUTER', 'THE RED DOG RUNS',
       'EATING APPLE PIE', 'THE ART OF WINNING');
    Pages : Array [1..4] of Integer = (400, 200, 150, 250);

  var
    BookTitle:   String[30];
    MP, Minutes: Integer;
    BookFound:   Boolean;
    I, Hours:    Integer;

begin
  Write ('Enter book title: ');           Readln (BookTitle);
  Write ('Enter rate (minutes/page): ');  Readln (MP);

  I := 0;  BookFound := False;
  repeat
    Inc(I);
    BookFound := BookTitle = Title[I];
  until (I > 4) or BookFound;

  Minutes := MP * Pages[I];
  Hours := Trunc(Minutes) div 60;
  Minutes := Minutes - Hours * 60;
  Write (Hours, ' HOURS  ', Trunc(Minutes), ' MINUTES');
end.
```

```
{2.1}
program Two1T87;
{ -- This program will rotate a string N times to the left. }
  var
    St:   String[10];
    L, N: Byte;

begin
  Write ('Enter string: ');  Readln (St);
  Write ('Enter N: ');       Readln (N);

  L := Length(St);
  N := N mod L;
  Write   (Copy (St, N+1, L-N));
  Writeln (Copy (St, 1, N));
end.


{2.2}
program Two2T87;
{ -- This program will determine the number of diskettes bought. }
  var
    Vers, Maxs, Wabs: Integer;

begin
  for Vers := 1 to 98 do
    for Maxs := 1 to 99 - Vers do begin
      Wabs := 100 - Maxs - Vers;
      if (Wabs > 0) and
      (Vers * 225 + Maxs * 297 + Wabs * 120 = 23607) then
        begin
          Writeln (Vers,' VERS  ', Maxs,' MAXS  ', WABS,' Wabs');
          Exit;
        end;
    end;
end.
```

```
{2.3}
program Two3T87;
{ -- This program will display a subset of random numbers. }
uses Crt;
  var
    SetOfNum: Array [1..16] of Integer;
    Nums:     Array [1..5]  of Integer;
    NewNum:   Boolean;
    Ch:       Char;
    Num, I, NumDisplayed, LastIndex:  Integer;

begin
  I := 0;
  repeat
    Inc(I);
    Write ('Enter list item: ');  Readln (SetOfNum[I]);
  until SetofNum[I] < 0;
  LastIndex := I - 1;

  Randomize;
  repeat
    NumDisplayed := 0;
    repeat
      repeat  { -- Get unique random number }
        Num := SetOfNum[ Random(LastIndex) + 1 ];
        NewNum := True;
        for I := 1 to NumDisplayed do
          if Num = Nums[I] then
            NewNum := False;
      until NewNum = True;

      Writeln (Num);
      Inc(NumDisplayed);
      Nums[NumDisplayed] := Num;
    until NumDisplayed = 5;

    Writeln ('PRESS ANY KEY');
    repeat until KeyPressed;
    Ch := ReadKey;
  until Ch = Chr(27);
end.
```

```
{2.4}
program Two4T87;
{ -- This program will display all partitioned sum of a number. }
  var
    Num, I, J: Byte;

begin
  Write ('Enter a number less than 20: ');  Readln (Num);
  for I := Num downto 1 do
    if Num mod I = 0 then begin
      Write (' ': 30 - (Num div I));
      Write (I);
      for J := 2 to Num div I do
        Write ('+', I);
      Writeln;
    end;
end.


{2.5}
program Two5T87;
{ -- This program will calculate the fractional value. }

  var
    St:          String[3];
    A:           Array [1..3] of Integer;
    Num, Den, I: Integer;

begin
  Write ('Enter word: ');  Readln (St);
  for I := 1 to 3 do
    A[I] := Ord(St[I]) - 64;
  Num := A[1] * A[2] + A[2] * A[3] + A[3] * A[1];
  Den := A[1] * A[2] * A[3];
  for I := Den downto 1 do
    if (Num mod I = 0) and (Den mod I = 0) then begin
      Writeln (Num div I, '/', Den div I);  Exit;
    end;
end.
```

```
{2.6}
program Two6T87;
{ -- This program will find a subset of integers. }
  var
    Item:              Array [1..8] of Integer;
    N, S, I, J, Sum, Temp, LastIndex: Integer;

begin
  I := 0;
  repeat
    Inc(I);
    Write ('Enter set item: ');  Readln (Item[I]);
  until Item[I] < 0;
  LastIndex := I - 1;
  Write ('Enter N: ');  Readln (N);
  Write ('Enter S: ');  Readln (S);

  for I := 1 to LastIndex - 1 do
    for J := I + 1 to LastIndex do
      if Item[I] > Item[J] then begin
        Temp := Item[I];  Item[I] := Item[J];  Item[J] := Temp;
      end;

  Sum := 0;
  for I := 1 to N do
    Sum := Sum + Item[I];

  If Sum <= S then
    begin
      Writeln ('YES');
      for I := 1 to N do
        Write (Item[I], '  ')
    end
  else
    Writeln ('NO');
end.
```

```
{2.7}
program Two7T87;
{ -- This program will determine if patterns are legal/illegal. }
  var
    St:       String[20];
    I, BA, A: Byte;
    Legal:    Boolean;

begin
  Write ('Enter pattern: ');  Readln (St);
  Legal := True;
  I := 1;

  if Copy(St, I, 1) <> 'A' then  { -- does not start with A }
    Legal := False
  else begin  { -- starts with A }
    Inc(I);
    while Copy (St, I, 2) = 'BA' do  { -- skip valid BA's }
      I := I + 2;

    A := I;  { -- A = position before finding trailing A's }
    while (I <= Length (St)) and Legal do begin
      if Copy(St, I, 1) <> 'A' then  { -- invalid trailing letter}
        Legal := False;
      Inc(I);
    end;
    if A = I then { -- no trailing A's }
      Legal := False;
  end;

  if not Legal then Write ('IL');
  Writeln ('LEGAL PATTERN');
end.

{2.8}
program Two8T87;
{ -- This program will find integers having F factors. }
  var
    I, J, M, N, F, NumF: Integer;

begin
  Write ('Enter M, N, F: ');  Readln (M, N, F);
  for I := M to N do begin
    NumF := 0;
    for J := 1 to Trunc(Sqrt(I)) do
      if I mod J = 0 then
        NumF := NumF + 2;
    if Sqrt(I) = Trunc(Sqrt(I)) then
      Dec(NumF);
    if NumF = F then
      Writeln (I);
  end;
end.
```

```
{2.9}
program Two9T87;
{ -- This program will alphabetize 5 words according to rules. }
  var
    Word:        Array [1..5]  of String[12];
    Word2:       Array [1..5]  of String[12];
    St:          Array [1..12] of String[1];
    Temp:        String[12];
    I, J, K, L:  Byte;

begin
  for I := 1 to 5 do begin
    Write ('Enter word ', I, ': ');  Readln (Word[I]);
    L := Length( Word[I] );
    for J := 1 to L do
      St[J] := Copy(Word[I], J, 1);
    { -- Alphabetize letters within word and make WORD2. }
    for J := 1 to L - 1 do
      for K := J + 1 to L do
        if St[J] > St[K] then begin
          Temp := St[J];  St[J] := St[K];  St[K] := Temp;
        end;
    Word2[I] := '';
    for J := 1 to L do
      Word2[I] := Word2[I] + St[J];
  end;

  { -- Alphabetize Words according to Word2. }
  for J := 1 to 4 do
    for K := J + 1 to 5 do
      if Word2[J] > Word2[K] then begin
      Temp := Word2[J];  Word2[J] := Word2[K];  Word2[K] := Temp;
      Temp := Word[J];   Word[J]  := Word[K];   Word[K]  := Temp;
    end;

  for I := 1 to 5 do
    Writeln (Word[I]);
end.
```

```
{2.10}
program Two10T87;
{ -- This program will produce a super-duper input routine. }
uses Crt;
  var
    Row, Col, Max, Tipe, InitCol: Byte;
    Ch:       Char;
    ValidCh:  Boolean;
    Entry:    String[20];

begin
  Write ('Enter ROW, COL: ');  Readln (Row, Col);
  Write ('Enter MAX: ');       Readln (Max);
  Write ('Enter TYPE: ');      Readln (Tipe);

  ClrScr;  Entry := '';  InitCol := Col;
  repeat
    GotoXY (Col, Row);
    repeat until KeyPressed;
    Ch := ReadKey;
    if Ch = Chr(8) then begin  { -- Backspace pressed }
      if Length(Entry) > 0 then begin
        Entry := Copy (Entry, 1, Length(Entry)-1);
        Dec(Col);
        GotoXY (Col, Row);  Write (' ');
      end
    end
    else begin
      ValidCh := Length(Entry) < Max;
      If ValidCh then
        Case Tipe of
          1:  if not (Ch in ['A'..'Z', ' ']) then
                ValidCh := False;
          2:  if not (Ch in ['0'..'9', '.']) then
                ValidCh := False;
          3:  begin
                if Col-InitCol in [2, 5] then
                  if Ch <> '-' then ValidCh := False
                  else
                else
                  if not (Ch in ['0'..'9']) then
                    ValidCh := False;
               end;
        end;

      if ValidCh then begin
        Write (Ch);
        Entry := Entry + Ch;
        Inc(Col);
      end;
    end;
  until Ch = Chr(13);
  GotoXY (InitCol, Row+2);  Writeln (Entry);
end.
```

```
{3.1}
program Thr1T87;
{ -- This program will determine if 2 words are closely spelled. }
  type
    String10 = String[10];
  var
    Word1, Word2:    String10;
    Close:           Boolean;
    Len1, Len2, Min: Byte;
    PosDif:          Byte;

function PositionDiffer ({using} Word1, Word2: String10;
                                  Min: Byte): {giving} Byte;
{ -- This function will find the first position that differs. }
  var
    I : Byte;

begin
    for I := 1 to Min do
      if Copy(Word1, I, 1) <> Copy(Word2, I, 1) then begin
        PositionDiffer := I;  Exit;
      end;
    PositionDiffer := Min + 1;
end;  { -- function }

begin
  Write ('Enter word 1: ');  Readln (Word1);
  Write ('Enter word 2: ');  Readln (Word2);
  Len1 := Length(Word1);
  Len2 := Length(Word2);

  Close := False;
  if Word1 = Word2 then  { -- Words are the same }
    Close := True
  else if Abs(Len1 - Len2) < 2 then begin  { -- Could be close }
    { -- Find first character that differs. }
    if Len1 < Len2 then
      Min := Len1
    else
      Min := Len2;
    PosDif := PositionDiffer (Word1, Word2, Min);
    If PosDif > Min then  { -- Close (Same, or differ by add/del)}
      Close := True
    else
      if Len1 = Len2 then  { -- Check if 1 letter changed/trans }
        begin
          if (PosDif < Len1) and
          (Copy(Word1, PosDif+1, 1) = Copy(Word2, PosDif, 1)) and
          (Copy(Word2, PosDif+1, 1) = Copy(Word1, PosDif, 1)) then
               Inc(PosDif);  { -- possible skip over }
          if Copy(Word1, PosDif+1, Len1 - PosDif + 1) =
             Copy(Word2, PosDif+1, Len2 - PosDif + 1) then
            Close := True;
        end
      else  { -- Lengths differ by 1, Check for insertion/delete }
```

```
        if Len2 < Len1 then begin
          if Copy (Word2, PosDif,   Len2 - PosDif + 1) =
             Copy (Word1, PosDif+1, Len1 - PosDif) then
            Close := True
          end
        else
          if Copy (Word1, PosDif,   Len1 - PosDif + 1) =
             Copy (Word2, PosDif+1, Len2 - PosDif) then
            Close := True;
    end;

  if Close then
    WriteLn ('CLOSE')
  else
    WriteLn ('NOT CLOSE');
end.
```

```
{3.2}
program Thr2T87;
{ -- This program will evaluate an NxN determinant for N=2,3,4. }
  var
    I, J, K:     Byte;
    A, B:        Array [1..4, 1..6] of Integer;
    Sum, Tot, N: Integer;
    Power:       Integer;

procedure EvaluateDetWithout ({using} K: Integer);
{ -- This procedure evaluates a 3 x 3 determinant w/o col K }
var
  I, J, S: Byte;

begin
  for I := 1 to 3 do begin
    S := 0;
    for J := 1 to 4 do
      if J <> K then begin  { -- Create an 3 row by 4 col array }
        Inc(S);
        B[I,S]   := A[I,J];
        B[I,S+3] := A[I,J];
      end;
  end;
  Sum := 0;
  for I := 1 to 3 do
    Sum := Sum + B[1,I]   * B[2,I+1] * B[3,I+2]
               - B[1,I+2] * B[2,I+1] * B[3,I];
end;

begin
  Write ('Enter dimension N: ');  Readln (N);
  for I := 1 to N do
    for J := 1 to N do begin
      Write ('Enter row ', I, ', col ', J, ': ');
```

```
      Readln (A[I,J]);
    end;

  if N = 2 then begin  { 2 x 2 determinant }
    Sum := A[1,1] * A[2,2] - A[1,2] * A[2,1];
    Writeln (Sum);
    end
  else if N = 3 then begin  { 3 x 3 determinant }
    EvaluateDetWithout (4);
    Writeln (Sum);
    end
  else begin
    Tot := 0;
    for K := 1 to 4 do begin
      EvaluateDetWithout (K);
      Power := 1;
      for I := 1 to K do
        Power := Power * (-1);
      Tot := Tot + Sum * A[4,K] * Power;
    end;
    WriteLn (Tot);
  end;
end.
```

```
{3.3}
program Thr3T87;
{ -- This program will display the number of word occurrences. }
  type
    String12 = String[12];
  var
    Lines:       String[255];
    Word:        Array [1..20] of String12;
    WordTot:     Array [1..20] of Byte;
    NextWord:    String12;
    NumOfWords:  Byte;
    NewWord:     Boolean;
    Start, I:    Byte;
    WordInd:     Byte;

function GetWord ({using} var Start: Byte): {giving} String12;
{ -- This procedure get the next word in the passage at Start. }
var
  I:         Byte;
  NextWord:  String12;
  Ch:        Char;
  EndOfWord: Boolean;

begin
  I := Start;  EndOfWord := False;  NextWord := '';
  repeat
    Ch := Lines[I];
    if Ch in ['A'..'Z', ''''] then
      NextWord := NextWord + Ch
    else
      EndOfWord := True;
    Inc(I);
  until (I > Length(Lines)) or EndOfWord;
  Start := I;  GetWord := NextWord;
end;

begin
  Write ('Enter text: ');  Readln (Lines);
  Start := 1;  NumOfWords := 0;
  repeat
    NextWord := GetWord(Start);
    if NextWord > '' then
      NewWord := True
    else
      NewWord := False;
    WordInd := 0;
    while (WordInd < NumOfWords) and NewWord do begin
      Inc(WordInd);
      if NextWord = Word[WordInd] then NewWord := False;
    end;
    if NewWord then begin  { -- Add new word to list of words }
      Inc(NumOfWords);
      Word[NumOfWords] := NextWord;
      WordTot[NumOfWords] := 1;
    end
```

```
    else  { -- Increment # of times this word appears }
      Inc( WordTot[WordInd] );
  until Start > Length(Lines);

  for I := 1 to NumOfWords do
    Writeln (WordTot[I], ' ', Word[I]);
end.
```

```
{3.4}
program Thr4T87;
{ -- This program will encrypt a string such that when this
  -- code is entered, the string will be reproduced. }
  var
    St:          String[50];
    I, NumOfCh: Byte;
    Result:      Integer;
    Ch, NextCh: Char;
    AscSt:       String[4];
    Asc:         Array [1..50] of Byte;
    CodeNum:     Byte;

begin
  Write ('Enter text: ');  Readln (St);
  NumOfCh := 0;  I := 1;
  while (I <= Length(St)) do begin
    Ch := St[I];  Inc(NumOfCh);
    if Ch = '\' then
      begin  { -- Either another / or ### follows }
        Inc(I);
        NextCh := St[I];
        if NextCh <> '\' then
          begin { -- Next 3 characters are the ASC code }
            AscSt := Copy (St, I, 3);
            Val (AscSt, Asc[NumOfCh], Result);
            I := I + 2;
          end
        else  { / follows }
          Asc[NumOfCh] := Ord(NextCh);
      end
    else  { -- A regular character }
      Asc[NumOfCh] := Ord(Ch);
    Inc(I);
  end;  { -- while I }

  { -- Encrypt code }
  for I := 1 to NumOfCh do begin
    CodeNum := 255 - Asc[I];
    If CodeNum in [32 .. 92] then begin
      Write (Char(CodeNum));
      if CodeNum = Ord('\') then
        Write ('\');
      end
    else  { -- Non printable }
      begin
        Str (1000 + CodeNum: 4, AscSt);
        Write ('\');  Write(Copy(AscSt, 2, 3));
      end;
  end;
end.
```

```
{3.5}
program Thr5T87;
{ -- This program will unscramble the numbers 5132, 4735, and
  -- 8014153 so that the first times the second equal the
  -- third with a missing digit }
  const
    A : Array [1..4] of Byte = (5, 1, 3, 2);
    B : Array [1..4] of Byte = (4, 7, 3, 5);
    C : Array [1..7] of Byte = (8, 0, 1, 4, 1, 5, 3);
  var
    I, J, K, L, Perm24:       Byte;
    Prod:                     LongInt;
    Result:                   Byte;
    ANum, BNum:               Array [1..24] of LongInt;
    St:                       String[8];
    PCh:                      Array [1..8] of Char;
    Match:                    Boolean;

begin
  { -- Generate 24 permuations of 5132 and 4735 each. }
  Perm24 := 0;
  for I := 1 to 4 do
    for J := 1 to 4 do
      for K := 1 to 4 do begin
        L := 4+3+2+1 -I-J-K;
        if (I=J) or (J=K) or (I=K) then { -- do nothing }
        else begin
          Inc(Perm24);
          ANum[Perm24] := A[I]*1000 + A[J]*100 + A[K]*10 + A[L];
          BNum[Perm24] := B[I]*1000 + B[J]*100 + B[K]*10 + B[L];
        end;
      end;  { -- for K }

  for I := 1 to 24 do
    for J := 1 to 24 do begin
      Prod := ANum[I] * BNum[J];
      if not (Prod < 10E6) then begin { -- has 8 digits }
        Str (Prod, St);
        for K := 1 to 8 do
          PCh[K] := St[K];
        L := 1;
        repeat
          Match := False;  K := 0;
          repeat
            Inc(K);
            if C[L] = Ord(PCh[K]) - Ord('0') then begin
              PCh[K] := ' ';
              Match := True;
            end
          until (K = 8) or Match;
          Inc(L);
        until (L > 7) or not Match;

        if Match then
          Writeln (ANum[I], '  ', BNum[J], '  ', St);
```

```
      end;  { -- if }
    end;  { -- for J }
end.



{3.6}
program Thr6T87;
{ -- This program will display the front colors on the Rubik's
  -- Pocket Cube after a move of T or F is performed. }
  const
    A : Array [1..24] of Char =
       ('W', 'W', 'W', 'W', 'Y', 'Y', 'Y', 'Y',
        'O', 'O', 'O', 'O', 'R', 'R', 'R', 'R',
        'G', 'G', 'G', 'G', 'B', 'B', 'B', 'B');
  var
    I, J:    Byte;
    Move, X: Char;

begin
  repeat
    Write ('Enter T, F, or Q: ');  Readln (Move);
    if Move = 'T' then
      begin
        X    := A[1];  A[1] := A[3];  A[3] := A[4];
        A[4] := A[2];  A[2] := X;
        X    := A[5];  A[5] := A[9];  A[9] := A[13];
        A[13]:= A[17]; A[17]:= X;
        X    := A[6];  A[6] := A[10]; A[10]:= A[14];
        A[14]:= A[18]; A[18]:= X;
      end
    else if Move = 'F' then
      begin
        X    := A[5];  A[5] := A[7];  A[7] := A[8];
        A[8] := A[6];  A[6] := X;
        X    := A[3];  A[3] := A[20]; A[20]:= A[22];
        A[22]:= A[9];  A[9] := X;
        X    := A[4];  A[4] := A[18]; A[18]:= A[21];
        A[21]:= A[11]; A[11]:= X;
      end;
    if Move <> 'Q' then begin
      Writeln (A[5], '  ', A[6]);
      Writeln (A[7], '  ', A[8]);
    end
  until Move = 'Q';
end.
```

```
{3.7}
program Thr7T87;
{ -- This program will simulate a drill of Adding Roman Numerals.}
uses Crt;
  const
    RN:  Array[1..7] of Char= ('M', 'D', 'C', 'L', 'X', 'V', 'I');
    RNV: Array[1..7] of Integer = (1000, 500, 100, 50, 10, 5, 1);
  var
    Option:       Byte;
    Name, Dayte:  String[8];

procedure Do3Problems;
{ -- This procedure will allow the user to do 3 addition problems}
  var
    I, J, K:      Byte;
    Right, Wrong: Byte;
    Prob, XX:     Byte;
    Num:          Array [1..3] of Byte;
    RNum:         Array [1..3] of String[12];
    Ans:          String[12];
    X:            Real;
    Miss:         Byte;
    L1, L2, Col:  Byte;
    Arabic:       Byte;
    Ri, Wr:       Array [1..3] of String[12];
    RiA:          Array [1..3] of Byte;

begin
  Right := 0;  Wrong := 0;
  for Prob := 1 to 3 do begin
    ClrScr;
    Randomize;
    Num[1] := Random(19) + 1;  Num[2] := Random(19) + 1;
    Num[3] := Num[1] + Num[2];  Arabic := Num[3];
    for K := 1 to 3 do
      RNum[K] := '';
    for K := 1 to 3 do
      for I := 1 to 7 do begin
        X := Num[K] / RNV[I];
        if (X < 2) and (X >= 9/5) and ((I=2) or (I=4) or (I=6))
        then { null }
        else begin
          XX := Trunc(X);
          If XX = 9 then
            RNum[K] := RNum[K] + RN[I] + RN[I-2]
          else if XX = 4 then
            RNum[K] := RNum[K] + RN[I] + RN[I-1]
          else if XX > 0 then
            for J := 1 to XX do
              RNum[K] := RNum[K] + RN[I];
          Num[K] := Num[K] - RNV[I] * XX;
        end;
      end; { -- for I }
```

```
    { -- Display Problem }
    GotoXY (15, 10);  Write (RNum[1]);
    L1 := Length(RNum[1]);  L2 := Length(RNum[2]);
    Col := 15 + (L1 - L2) - 2;
    GotoXY (Col, 11);  Write ('+ ', RNum[2]);
    GotoXY (Col, 12);
    for I := 1 to 2 + L2 do Write ('-');
    Miss := 0;
    repeat
      GotoXY (Col, 13);  Readln (Ans);

      { -- Evaluate Answer }
      if Ans = RNum[3] then begin
        Inc(Right);  Miss := 0;  end
      else  { -- Incorrect answer }
        if Miss > 0 then begin  { -- Second Miss }
          Miss := 0;  Sound (400); Delay (200);  NoSound;
          Inc(Wrong);  Wr[Wrong] := Ans;
          Ri[Wrong] := RNum[3];  RiA[Wrong] := Arabic;
          end
        else begin  { -- First Miss }
          Miss := 1;  Sound (400);  Delay (200);  NoSound;
          GotoXY (Col, 16);  Write (Arabic);
          GotoXY (Col, 13);  ClrEol;
        end;
    until Miss = 0;
  end;  { -- for Prob }

  { -- Progress Report }
  ClrScr;  GotoXY (11,1);  Writeln ('PROGRESS REPORT');
  Writeln ('DATE: ', Dayte);
  Writeln ('NAME: ', Name);
  Writeln ('NUMBER CORRECT: ', Right);
  Writeln ('NUMBER OF EXERCISES: 3');
  Writeln ('PERCENT CORRECT: ', Round(RIGHT / 3 * 100), '%');
  Writeln;
  if Wrong > 0 then begin
    GotoXY (1, 15);
    Writeln ('WRONG ANSWER   CORRECT ANSWER  ARABIC');
    for I := 1 to Wrong do begin
      GotoXY (1, 16+I);  Write (Wr[I]);
      GotoXY (16, 16+I); Write (Ri[I]);
      GotoXY (32, 16+I); Write (RiA[I]);
    end;
    GotoXY (1, 23);  Writeln ('PRESS ANY KEY TO RETURN TO MENU.');
    repeat until KeyPressed;
  end;
end;

begin
  Write ('Enter name: ');  Readln (Name);
  Write ('Enter date: ');  Readln (Dayte);

  repeat
    ClrScr;
```

```
    Writeln ('1. INSTRUCTION PAGE');
    Writeln ('2. PRACTICE 3 PROBLEMS');
    Writeln ('3. QUIT');
    Readln (Option);
    if Option = 1 then  { -- Display instructions }
      begin
        ClrScr;
        Writeln ('YOU WILL BE GIVEN 3 PROBLEMS TO');
        Writeln ('WORK. A PROBLEM WILL CONSIST OF');
        Writeln ('ADDING TWO RANDOMLY GENERATED');
        Writeln ('ROMAN NUMERALS LESS THAN 20.');
        Writeln ('YOU WILL TYPE YOUR ANSWER IN');
        Writeln ('ROMAN NUMERALS AND PRESS ''RETURN.''');
        Writeln ('(PRESS ANY KEY TO RETURN TO MENU.)');
        repeat until KeyPressed;
      end
    else if Option = 2 then  { -- Practice 3 problems }
      Do3Problems;
  until Option = 3;
end.
```

```
{3.8}
program Thr8T87;
{ -- This program will determine the area shared w/2 rectangles. }
  var
    A, B, X, Y:       Array [1..4] of Integer;
    AB, XY:           Array [0..20, 0..20] of Integer;
    I, J, Width,
    Width2, Height:  Integer;

begin
  for I := 1 to 4 do begin
    Write ('Enter X,Y: ');  Readln (X[I], Y[I]);
    X[I] := Abs(X[I]);  Y[I] := Abs(Y[I]);
  end;
  Writeln;
  for I := 1 to 4 do begin
    Write ('Enter A,B: ');  Readln (A[I], B[I]);
    A[I] := Abs(A[I]);  B[I] := Abs(B[I]);
  end;

  { -- Initialize AB and XY arrays }
  for I := 0 to 20 do
    for J := 0 to 20 do begin
      AB[I,J] := 0;  XY[I,J] := 0;  end;

  { -- Store a 1 in each occupied square }
  for I := A[1] to A[2] do
    for J := B[4] to B[1] do
      AB[I, J] := 1;

  { -- Determine area in common (height-1 x Width-1) }
  Width := 0;  Height := 0;
  for I := X[1] to X[2] do begin
    for J := Y[4] to Y[1] do
      if (AB[I, J] = 1) then
        Inc(Width);
    if Width > 0 then begin
      Inc(Height);  Width2 := Width;  Width := 0;
    end;
  end;

  Writeln ((Height -1) * (Width2 -1));
end.
```

```
{3.9}
program Thr9T87;
{ -- This program will divide 2 big numbers w/at most 30 digits. }
  var
    ASt, BSt:            String[30];
    A, B:                Array[1..30] of Integer;
    Ch:                  Char;
    LenA, LenB, Quot:    Byte;
    I:                   Integer;
    LastAPos, LastAInd:  Byte;
    DigitsAdded:         Byte;
    AtLeast1Divide:      Boolean;

function ALessThanB: {giving} Boolean;
{ -- This function returns true if A[..] is less than B[..] }
  var
    I: Byte;

begin
  if LastAInd > LenB then
    ALessThanB := False
  else if LastAInd < LenB then
    ALessThanB := True
  else begin  { -- both A and B are same length }
    I := LenB;
    while (I > 1) and (A[I] = B[I]) do
      Dec(I);
    if A[I] < B[I] then  { -- Found position where A is < B }
      ALessThanB := True
    else
      ALessThanB := False;
  end;
end;

procedure AttachDigitToA;
{ -- This procedure will attach another digit at end of A[..] }
begin
  for I := LastAInd downto 1 do
    A[I+1] := A[I];
  if A[LastAInd+1] > 0 then
    Inc(LastAInd);
  Inc(LastAPos);
  Ch := ASt[LastAPos];
  A[1] := Ord(Ch) - Ord('0');
end;

procedure Sub_B_From_A;
{ -- This procedure will subtract B[..] from A[..] with borrowing}
  var
    Borrow: Byte;

begin
  for I := 1 to LenB do begin
    if B[I] <= A[I] then Borrow := 0
      else begin
```

```
        Borrow := 10;
        Dec(A[I+1]);
      end;
    A[I] := A[I] - B[I] + Borrow;
  end;
  { -- Find first non-zero of A[] for LastAInd }
  while (LastAInd > 1) and (A[LastAInd] = 0) do
    Dec(LastAInd);
end;

procedure DivideAbyB;
{ -- This procedure will divide A[..] by B[..] and display quot. }
begin
  Quot := 1;
  while not ALessThanB and (Quot < 10) do begin
    Sub_B_From_A;  Inc(Quot);
  end;
  Write (Quot - 1);
end;
      { -- Main program routine }
begin
  Write ('Enter first number: ');   Readln (ASt);
  Write ('Enter second number: ');  Readln (BSt);
  LenA := Length (ASt);  LenB := Length (BSt);  { -- LenA > LenB }

  { -- Store B number in Array: 456 becomes B[3]=6,B[2]=5,B[1]=4 }
  for I := LenB downto 1 do begin
    Ch := BSt[I];
    B[LenB-I+1] := Ord(Ch) - Ord('0');
  end;
  { -- Store equal number of digits in A as was in B }
  if LenB <= LenA then LastAPos := LenB
    else LastAPos := Length(ASt);
  for I := LastAPos downto 1 do begin
    Ch := ASt[I];
    A[LastAPos-I+1] := Ord(Ch) - Ord('0');
  end;
  LastAInd := LastAPos;
  if ALessThanB and (LastAPos < LenA) then
    { -- Attach 1 more digit so A > B }
    AttachDigitToA;
  AtLeast1Divide := False;

  { -- Perform systematic division by attaching digits
    -- until no more digits }
  while (LastAPos < LenA) or not ALessThanB do begin
    DigitsAdded := 0;
    while ALessThanB and (LastAPos < LenA) do begin
      AttachDigitToA;  Inc(DigitsAdded);
    end;
    for I := 1 to DigitsAdded-1 do
      { -- Print 0's for each excessive digit }
      Write ('0');
    DivideAbyB;
    AtLeast1Divide := True;
```

```
  end;  { -- while }

  { -- Display Remainder }
  if not AtLeast1Divide then Write ('0');  { -- No quotient, A<B }
  Write (' REMAINDER ');
  for I := LastAInd downto 1 do
    Write (A[I]);
end.



{3.10}
program Thr10T87;
{ -- This program will generate random mazes with 8 x 5 paths. }
uses Crt;
  var
    { -- A = Forbidden segments,  PointUsed = Existing points }
    A:         Array [0..33, 0..33] of  Byte;
    PointUsed: Array [0..33, 0..33] of  Byte;
    L, W, Linc, Winc, Lnum, Wnum:       Byte;
    NumOfLines, D, X, Y, X2, Y2, I, J:  Byte;
    LinesDrawn, NumofTries:             Byte;
    SegmentDrawn:                       Boolean;

begin
  ClrScr;  Randomize;  L := 8;  W := 5;
  NumOfLines := (L-1) * (W-1);
  LinesDrawn := 0;
  Linc := 32 div L;  Winc := 15 div W;
  Lnum := L;  Wnum := W;
  for I := 0 to 33 do
    for J := 0 to 33 do begin
      PointUsed[I, J] := 0;  A[I, J] := 0;
    end;

  { -- Draw perimeter }
  for I := 1 to 33 do begin
    Write ('*');  PointUsed[I-1, 0] := 1;
  end;
  for I := 1 to 14 do begin
    GotoXY (1, I+1);  Write ('*');  PointUsed[0, I] := 1;
    GotoXY (33,I+1);  Write ('*');  PointUsed[L, I] := 1;
  end;  Writeln;
  for I := 1 to 33 do begin
    Write ('*');  PointUsed[I-1, W] := 1;
  end;

  A[0, 0] := 1;  A[Lnum, 0] := 1;  A[Lnum, Wnum] := 1;
  A[0, Wnum] := 1;
  repeat
    { -- Get point that exist but is not forbidden }
    repeat
      X := Random(Lnum*2) - (Lnum div 2);
      Y := Random(Wnum*2) - (Wnum div 2);
```

```
    if X < 0 then X := 0;
    if X > Lnum then X := Lnum;
    if Y < 0 then Y := 0;
    if Y > Wnum then Y := Wnum;
  until (PointUsed[X, Y] = 1) and (A[X, Y] = 0);

  repeat
    D := Random(4);  { -- Random direction }

    SegmentDrawn := False;  NumOfTries := 0;
    repeat
      NumOfTries := NumOfTries + 1;
      Inc(D);  If D > 4 then D := D - 4;
      Case D of
        1: begin  { -- Up }
             if (Y > 0) and not (PointUsed[X, Y-1] = 1) then
             begin
               for J := 0 to Winc - 1 do begin
                 GotoXY (X*Linc+1, Y*Winc-J);  Write ('*');
               end;
               X2 := X;  Y2 := Y - 1;
               SegmentDrawn := True;
             end;
           end;

        2: begin  { -- Right }
             if (X < LNum) and not (PointUsed[X+1, Y] = 1) then
             begin
               for J := 0 to Linc - 1 do begin
                 GotoXY (X*Linc+2+J, Y*Winc+1);  Write ('*');
               end;
               X2 := X + 1;  Y2 := Y;
               SegmentDrawn := True;
             end;
           end;

        3: begin  { -- Down }
             if (Y < Wnum) and not (PointUsed[X, Y+1] = 1) then
             begin
               for J := 0 to Winc - 1 do begin
                 GotoXY (X*Linc+1, Y*Winc+2+J);  Write ('*');
               end;
               X2 := X;  Y2 := Y + 1;
               SegmentDrawn := True;
             end;
           end;

        4: begin  { -- Left }
             if (X > 0) and not (PointUsed[X-1, Y] = 1) then
             begin
               for J := 0 to Linc - 1 do begin
                 GotoXY (X*Linc-J, Y*Winc+1);  Write ('*');
               end;
               X2 := X - 1;  Y2 := Y;
               SegmentDrawn := True;
```

```
              end;
            end;
        end;  { -- case }
      until SegmentDrawn or (NumofTries = 4);

      if SegmentDrawn then begin
        PointUsed[X2, Y2] := 1;
        Inc(LinesDrawn);
        X := X2;  Y := Y2;
        end
      else  { -- No more segments can be drawn from this point }
        A[X, Y] := 1;
    until (LinesDrawn = NumOfLines) or not SegmentDrawn;
  until (LinesDrawn = NumOfLines);  { -- Get new point of
                                       -- Segment not drawn }
  { -- Open doors }
  X := Random(Wnum) + 1;  Y := Random (Wnum) + 1;
  for J := 0 to Winc - 2 do begin
    GotoXY (1,  X * Winc - J);  Write (' ');
    GotoXY (33, Y * Winc - J);  Write (' ');
  end;
  GotoXY (1, 23);

end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '88 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T88;
{ -- This program clears the screen and prints a phrase 10 times.}
uses Crt;
  var
    I: Byte;

begin
  ClrScr;
  for I := 1 to 10 do
    Writeln ('THE BEST COMPUTER CONTEST!');
end.


{1.2}
program One2T88;
{ -- This program determines if a given input is integer or real.}
  var
    Num: Real;

begin
  Write ('Enter #: ');  Readln (Num);
  if Trunc(Num) - Num = 0 then
    Writeln ('INTEGER')
  else
    Writeln ('REAL');
end.


{1.3}
program One3T88;
{ -- This program calculates the number of bytes on N diskettes. }
  var
    N, Bytes: LongInt;

begin
  Write ('Enter N: ');  Readln (N);
  Bytes := N * 40 * 8 * 512;
  Writeln (Bytes);
end.
```

```
{1.4}
program One4T88;
{ -- This program prints the computer component missing. }
  const
    Comp: Array[1..5] of String[9] =
          ('CPU', 'PRIMARY', 'SECONDARY', 'INPUT', 'OUTPUT');
  var
    A:          String[9];
    I, J, Sum: Byte;

begin
  Sum := 0;
  for I := 1 to 4 do begin
    Write ('Enter component: ');  Readln (A);
    for J := 1 to 5 do
      if A = Comp[J] then Sum := Sum + J
  end;
  { -- The missing index = (1+2+3+4+5) - Sum }
  Writeln (Comp[15 - Sum]);
end.


{1.5}
program One5T88;
{ -- This program displays 4 rectangles of asterisks with #s. }
uses Crt;
  var
    I: Byte;

begin
  ClrScr;
  for I := 1 to 79 do
    Write ('*');

  for I := 2 to 24 do begin
    GotoXY (1,I);  Write ('*');
    GotoXY (40,I); Write ('*');
    GotoXY (79,I); Write ('*');
  end;

  for I := 1 to 79 do begin
    GotoXY (I,12); Write ('*');
  end;

  for I := 1 to 79 do begin
    GotoXY (I,24); Write ('*');
  end;

  GotoXY (20,6);  Write (1);
  GotoXY (60,6);  Write (2);
  GotoXY (20,18); Write (3);
  GotoXY (60,18); Write (4);
end.
```

```
{1.6}
program One6T88;
{ -- This program displays the acronym for a given set of words. }
  var
    I:  Byte;
    St: String[80];

begin
  Write ('Enter words: ');  Readln (St);
  Write (Copy(St, 1, 1));

  for I := 2 to Length(St) do begin
    if Copy(St, I, 1) = ' ' then
      Write (Copy(St, I+1, 1));
  end;
end.


{1.7}
program One7T88;
{ -- This program will display 3 computer names in order of size.}
  var
    N1, N2, N3, T1, T2, T3: String[10];

begin
  Write ('Enter name: ');  Readln (N1);
  Write ('Enter type: ');  Readln (T1);
  Write ('Enter name: ');  Readln (N2);
  Write ('Enter type: ');  Readln (T2);
  Write ('Enter name: ');  Readln (N3);
  Write ('Enter type: ');  Readln (T3);
  Writeln;

  if T1 = 'MICRO' then
    Writeln (N1)
  else if T2 = 'MICRO' then
    Writeln (N2)
  else
    Writeln (N3);

  if T1 = 'MINI' then
    Writeln (N1)
  else if T2 = 'MINI' then
    Writeln (N2)
  else
    Writeln (N3);

  if T1 = 'MAINFRAME' then
    Writeln (N1)
  else if T2 = 'MAINFRAME' then
    Writeln (N2)
  else
    Writeln (N3);
end.
```

```
{1.8}
program One8T88;
{ -- This program will count the number of cans to be stacked. }
  var
    N, Cans, Sum: Integer;

begin
  Write ('Enter N: ');  Readln (N);
  Cans := N;  Sum := 0;
  while (Cans > 0) do begin
    Sum := Sum + Cans;
    Cans := Cans - 2;
  end;
  Writeln (Sum);
end.


{1.9}
program One9T88;
{ -- This program simulates a queue w/options: ADD, TAKE, QUIT. }
  var
    Min, Max: Integer;
    Command:  String[4];
    A:        Array [1..10] of Integer;

begin
  Min := 0;
  Max := 0;
  repeat
    Write ('Enter command: ');  Readln (Command);
    if Command = 'ADD' then
      begin
        Inc(Max);
        Write ('Enter integer: ');  Readln (A[Max]);
      end
    else if Command = 'TAKE' then
      begin
        Inc(Min);
        Writeln (A[Min]);
      end
  until Command = 'QUIT';
end.
```

```
{1.10}
program One10T88;
{ -- This program determines events of history between dates. }
  type
    Ar = Array [1..7] of String[30];
  const
    Date: Array [1..7] of Integer =
              (1642, 1801, 1830, 1890, 1944, 1946, 1949);
    Per: Ar = ('BLAISE PASCAL', 'JOSEPH JACQUARD',
               'CHARLES BABBAGE', 'HERMAN HOLLERITH',
              'HOWARD AIKEN', 'ECKERT AND MAUCHLY', 'VON NEUMAN');
    Inv: Ar = ('ADDING MACHINE', 'PUNCHCARD AND WEAVING LOOM',
               'DESIGN OF ANALYTIC ENGINE',
               'PUNCHCARD TABULATING MACHINE', 'MARK I',
               'ENIAC', 'EDVAC');
  var
    Y1, Y2, I: Integer;

begin
  Write ('Enter years: ');  Readln (Y1, Y2);
  for I := 1 to 7 do begin
    if (Date[I] >= Y1) and (Date[I] <= Y2) then
      Writeln (Per[I], ' INVENTED ', Inv[I]);
  end;
end.
```

```
{2.1}
program Two1T88;
{ -- This program displays a solid diamond of asterisks. }
uses Crt;
  var
    I, J, N, NumOfSpaces: Integer;

begin
  Write ('Enter N: ');  Readln (N);

  { -- Display top half of diamond. }
  I := 1;
  repeat
    NumOfSpaces := (N - I) div 2 + 1;
    Write (' ': NumOfSpaces);
    for J := 1 to I do
      Write ('*');
    Writeln;
    I := I + 2;
  until I = N;
  I := I + 2;

  { -- Display middle row and bottom half of diamond. }
  repeat
    I := I - 2;
    NumOfSpaces := (N - I) div 2 + 1;
    Write (' ': NumOfSpaces);
    for J := 1 to I do
      Write ('*');
    Writeln;
  until I = 1;
end.
```

```
{2.2}
program Two2T88;
{ -- This program determines the efficiency order of 3 sorts. }
  const
    BS = 'BUBBLE SORT';
    SS = 'SHELL SORT';
    QS = 'QUICK SORT';
  var
    N:       Integer;
    B, S, Q: Real;

begin
  Write ('Enter N: ');  Readln (N);
  B := N * (N - 1) / 2;
  S :=     (Ln(N) / Ln(2));  S := N * S * S;
  Q := N * (Ln(N) / Ln(2));

  if (B < S) and (B < Q) then
    begin
      Writeln (BS);
      if S < Q then
        begin
          Writeln (SS);  Writeln (QS);
        end
      else
        begin
          Writeln (QS);  Writeln (SS);
        end
    end
  else if (S < B) and (S < Q) then
    begin
      Writeln (SS);
      if B < Q then
        begin
          Writeln (BS);  Writeln (QS);
        end
      else
        begin
          Writeln (QS);  Writeln (BS);
        end
    end
  else  { -- Q is less than both S and B }
    begin
      Writeln (QS);
      if B < S then
        begin
          Writeln (BS);  Writeln (SS);
        end
      else
        begin
          Writeln (SS);  Writeln (BS);
        end
    end
end.
```

```
{2.3}
program Two3T88;
{ -- This program determines the number of people in a group. }
  type
    Ar = Array [1..4] of Byte;
  const
    Di: Ar = (2, 3, 5, 7);
    Re: Ar = (1, 2, 1, 2);
  var
    Num, I: Byte;
    Found:  Boolean;

begin
  Num := 1;
  repeat
    Inc(Num);
    Found := True;
    for I := 1 to 4 do
      if (Num mod Di[I]) <> Re[I] then
        Found := False;
  until Found or (Num > 200);
  Writeln (Num);
end.


{2.4}
program Two4T88;
{ -- This program generates 5 random numbers between 0 and 9999. }
  const
    EightDigits = 10E7;
  var
    I, J:       Byte;
    Seed, Prod: LongInt;
    St, SeedSt: String[8];
    Code:       Integer;

begin
  Write ('Enter seed: ');  Readln (Seed);
  for I := 1 to 5 do begin
    Prod := Seed * Seed;
    while (Prod < EightDigits) and (Prod <> 0) do
      Prod := Prod * 10;
    Str (Prod, St);
    SeedSt := Copy (St, 3, 4);
    Val (SeedSt, Seed, Code);
    Writeln (Seed);
  end;
end.
```

```
{2.5}
program Two5T88;
{ -- This program checks to see if data transmitted is Correct. }
  var
    Bit, Par: String[8];
    I, One:   Byte;
    Error:    Boolean;

begin
  Write ('Enter bits: ');  Readln (Bit);
  Write ('Enter parity: ');  Readln (Par);
  if Length(Bit) < 8 then
    Writeln ('ERROR')
  else
    begin
      Error := False;
      One := 0;
      for I := 1 to 8 do begin
        If not (Bit[I] in ['0','1']) then
          Error := True;
        If Bit[I] = '1' then
          Inc(One);
      end;  { -- for }

      if (One mod 2 = 0) and (Par <> 'EVEN') then
        Error := True
      else if ((One mod 2) <> 0) and (Par <> 'ODD') then
        Error := True;
      if Error then
        Writeln ('ERROR')
      else
        WriteLn ('CORRECT');
    end;  { -- else }
end.
```

```
{2.6}
program Two6T88;
{ -- This program will calculate the area of a polygon. }
  var
    I, N: Byte;
    X, Y: Array [1..10] of Integer;
    Sum:  Integer;

begin
  Write ('Enter n: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter vertex: ');
    Readln (X[I], Y[I]);
  end;

  Sum    := 0;
  X[N+1] := X[1];
  Y[N+1] := Y[1];
  for I := 1 to N do
    Sum := Sum + X[I] * Y[I+1] - Y[I] * X[I+1];
  Writeln ('AREA = ',Abs(Sum) / 2 : 4:1);
end.
```

```
{2.7}
program Two7T88;
{ -- This program displays the date before/after a given date. }
  const
     Mo: Array [1..12] of Byte =
         (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
  var
    Month, Day, D1, D2, M1, M2, Leap1, Leap2: Byte;
    Year, Y1, Y2:                             Integer;

begin
  Write ('Enter month, day, year: ');  Readln (Month, Day, Year);

  D1 := Day - 1;  D2 := Day + 1;
  M1 := Month;    M2 := Month;
  Y1 := Year;     Y2 := Year;
  Leap1 := 0;  Leap2 := 0;
  if (Y1 mod 4 = 0) and (Y1 mod 100 <> 0) then
    if (M1 = 3) and (D1 = 0) then
      Leap1 := 1
    else
      if (M2 = 2) and (D2 = 29) then
        Leap2 := 1;

  if D1 <= 0 then begin
    Dec(M1);
    if M1 > 0 then
      D1 := Mo[M1] + Leap1
    else
      begin
        M1 := 12;  D1 := Mo[M1];  Dec(Y1);
      end;
  end;  { -- If then }
  if D2 > (Mo[M2] + Leap2) then begin
    Inc(M2);  D2 := 1;
    if M2 > 12 then begin
      M2 := 1;  Inc(Y2);
    end;
  end;  { -- if then }

  Writeln (M1, '-', D1, '-', Y1);
  Writeln (M2, '-', D2, '-', Y2);
end.
```

```
{2.8}
program Two8T88;
{ -- This program displays a student's Cumulative G. P. Ave. }
  var
    Sem, Total, HrsTot:  Byte;
    Gr:                  Char;
    Hrs, Poynts, I:      Byte;
    CumTotal, CumHrs:    Byte;
    GPA, CGPA, LastCGPA: Real;
    Dismissed:           Boolean;

begin
  Sem := 1;  Dismissed := False;  LastCGPA := 0;
  CumHrs :=0; CumTotal :=0;
  while (Sem <= 8) and not Dismissed do begin
    Total := 0;  HrsTot := 0;
    for I := 1 to 4 do begin
      Write ('Enter grade, credits: ');
      Readln (Gr, Hrs);
      if Gr = 'F' then Gr := 'E';
      Poynts := 4 - (Ord(Gr) - 65);  { -- A=4,B=3,C=2,D=1,F=0 }
      Total  := Total  + Poynts * Hrs;
      HrsTot := HrsTot + Hrs;
    end;  { -- for }

    GPA := Total / HrsTot;
    GPA := Int (GPA * 1000 + 0.5) / 1000;
    Writeln (' GPA= ', GPA: 5: 3);
    CumTotal := CumTotal + Total;
    CumHrs   := CumHrs   + HrsTot;
    CGPA := CumTotal / CumHrs;
    CGPA := Int (CGPA * 1000 + 0.5) /1000;
    Writeln ('CGPA= ', CGPA: 5: 3);
    if CGPA < 1 then
      Dismissed := True;
    if (CGPA < 2) and (LastCGPA < 2) and (Sem > 1) then
      Dismissed := True;
    LastCGPA := CGPA;
    Inc(Sem);
  end;  { -- while }
  If Dismissed then
    Writeln ('STUDENT IS DISMISSED');
end.
```

```
{2.9}
program Two9T88;
{ -- This program displays 2 elements that form a battery. }
uses Crt;
  const
    Elem: Array [1..10] of String[8] =
     ('LITHIUM ', 'SODIUM  ', 'ZINC    ', 'IRON    ', 'TIN     ',
      'IODINE  ', 'SILVER  ', 'MERCURY ', 'BROMINE ', 'CHLORINE');
    Pot:  Array [1..10] of Real =
     ( +3.05, +2.71, +0.76, +0.44, +0.14,
       -0.54, -0.80, -0.85, -1.09, -1.36);

  var
    I, J, Count:    Byte;
    Dif, Volt, Tol: Real;
    Displayed:      Boolean;
    Ch:             String[1];

begin

  Write ('Enter Desired Voltage, Tolerance: ');
  Readln (Volt, Tol);

  Displayed := False;  Count := 0;
  for I := 1 to 10 do
    for J := 1 to 10 do begin
      Dif := Pot[I] - Pot[J];
      If (Dif >= Volt - Tol) and (Dif <= Volt + Tol) then begin
        Inc(Count);
        if (Count = 1) and Displayed then begin
          Writeln ('PRESS ANY KEY FOR MORE');
          Ch:= '';  While Ch = '' do Ch := ReadKey;
          Writeln;
        end;
        Writeln (Elem[I], '   ', Elem[J], '   ', Dif: 3: 2);
        Displayed := True;
      end;  { -- if Dif }
      if Count = 8 then begin
        Writeln;
        Count := 0;
      end;
    end;  { -- for J }
  if not Displayed then
    Writeln ('NO BATTERY CAN BE FORMED');
end.
```

```
{2.10}
program Two1088;
{ -- This program will keep score for a double dual race. }
uses Crt;
  var
    Init:                Array [1..21] of Char;
    TeamName:            Array [1..3] of Char;
    I, J, K:             Byte;
    StillUnique:         Boolean;
    UniqueTeams, Pl:     Byte;
    Team1Pos, Team2Pos: Array [1..7] of Byte;
    Team1,    Team2:     Byte;
    Team1Pl,  Team2Pl:   Byte;

begin
  ClrScr;  UniqueTeams := 0;
  for I := 1 to 21 do begin
    Write ('Place ', I: 2, ': ');  Readln (Init[I]);
    J := 0;  StillUnique := True;
    while (J < UniqueTeams) and StillUnique and (I > 1) do begin
      Inc(J);
      if TeamName[J] = Init[I] then
        StillUnique := False;
    end; { -- while }
    if StillUnique then
      begin
        Inc(UniqueTeams);
        TeamName[UniqueTeams] := Init[I];
      end;
  end; { -- for I }
  { -- Assert that Team[1,2,3] = 3 unique team Initials. }

  for I := 1 to 2 do
    for J := I+1 to 3 do begin
      PL := 0;  Team1 := 0;  Team2 := 0;
      Team1Pl := 0;  Team2Pl :=0;
      for K := 1 to 21 do begin
        if Init[K] = TeamName[I] then
        begin
          Inc(Pl);
          Team1 := Team1 + Pl;
          Inc(Team1Pl);
          Team1Pos[Team1Pl] := Pl
        end;
        if Init[K] = TeamName[J] then
        begin
          Inc(Pl);
          Team2 := Team2 + Pl;
          Inc(Team2Pl);
          Team2Pos[Team2Pl] := Pl
        end;
      end;  { -- for K }
      Team1 := Team1 - Team1Pos[6] - Team1Pos[7];
      Team2 := Team2 - Team2Pos[6] - Team2Pos[7];
      Writeln ('TEAM ', TeamName[I], ': ', Team1, ' POINTS');
```

```
      Writeln ('TEAM ', TeamName[J], ': ', Team2, ' POINTS');
      if (Team1 < Team2)
      or ((Team1 = Team2) and (Team1Pos[6] < Team2Pos[6])) then
        Write ('TEAM ', TeamName[I])
      else
        Write ('TEAM ', TeamName[J]);
      Writeln (' WINS!');  Writeln;
    end;  { -- for J }
end.
```

```
{3.1}
program Thr1T88;
{ -- This program puts a set of real numbers in numerical order. }
  const
    Order:    Array [0..9] of Byte = (0,8,1,2,5,4,3,9,7,6);
  var
    I, J, N:  Byte;
    A:        Array [1..10] of String[18];
    B:        Array [1..10] of Real;
    Temp:     Real;
    TempSt,
    Num:      String[18];
    NumVal,
    NumVal2:  Integer;
    Md:       Char;
    NumValSt: String[1];
    Result:   Integer;

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter #: ');  Readln (A[I]);
  end;

  { -- Replace digits in duplicated number }
  for I := 1 to N do begin
    Num := A[I];
    for J := 1 to Length(Num) do begin
      Md := Num[J];
      NumVal := Ord(Md) - Ord('0');
      if (NumVal > 0) or (Md = '0') then begin
        NumVal2 := Order[NumVal];
        Delete (Num, J, 1);
        Str (NumVal2, NumValSt);
        Insert (NumValSt, Num, J);
      end;
    end;  { -- for J }
    Val (Num, B[I], Result);
  end;  { -- for I }

{ -- Sort according to numbers with replaced digits }
  for I := 1 to N - 1 do
    for J := I + 1 to N do
      if B[I] > B[J] then begin
        Temp   := B[I];  B[I] := B[J];  B[J] := Temp;
        TempSt := A[I];  A[I] := A[J];  A[J] := TempSt;
      end;

  for I := 1 to N do
    Writeln (A[I]);
end.
```

```
{3.2}
program Thr2T88;
{ -- This program displays total number of ways to make change. }
  var
    Amount:           Real;
    MaxQ, MaxD, MaxN: Integer;
    Q, D, N, Count:   Integer;

begin
  Write ('Enter AMOUNT: ');  Readln (Amount);
  MaxQ := Trunc(Amount * 4);
  MaxD := Trunc(Amount * 10);
  MaxN := Trunc(Amount * 20);
  Count := 0;
  for Q := 0 to MaxQ do
    for D := 0 to MaxD - Trunc(2.5 * Q) do
      for N := 0 to MaxN - 5*Q - 2*D do
        Inc(Count);
  Writeln (Count);
end.
```

```
{3.3}
program Thr3T88;
{ -- This program determines if a point/box is inside a 2nd box. }

function Min (A: Real;  B: Real): Real;
begin
  if A < B then
    Min := A
  else
    Min := B;
end;

function Max (A: Real;  B: Real): Real;
begin
  if A > B then
    Max := A
  else
    Max := B;
end;

{ -- Start of Main Program }
  var
    PX, PY, PZ,
    C1X1, C1Y1, C1Z1, C1X2, C1Y2, C1Z2,
    C2X1, C2Y1, C2Z1, C2X2, C2Y2, C2Z2,
    C1MinX, C1MinY, C1MinZ, C1MaxX, C1MaxY, C1MaxZ,
    C2MinX, C2MinY, C2MinZ, C2MaxX, C2MaxY, C2MaxZ:  Real;
```

```
begin
  Write ('Enter point: ');  Readln (PX, PY, PZ);
  Write ('Enter cube1 diagonal point1: ');
    Readln (C1X1, C1Y1, C1Z1);
  Write ('Enter cube1 diagonal point2: ');
    Readln (C1X2, C1Y2, C1Z2);
  Write ('Enter cube2 diagonal point1: ');
    Readln (C2X1, C2Y1, C2Z1);
  Write ('Enter cube2 diagonal point2: ');
    Readln (C2X2, C2Y2, C2Z2);

  C1MinX := Min (C1X1, C1X2);
  C1MinY := Min (C1Y1, C1Y2);
  C1MinZ := Min (C1Z1, C1Z2);
  C2MinX := Min (C2X1, C2X2);
  C2MinY := Min (C2Y1, C2Y2);
  C2MinZ := Min (C2Z1, C2Z2);
  C1MaxX := Max (C1X1, C1X2);
  C1MaxY := Max (C1Y1, C1Y2);
  C1MaxZ := Max (C1Z1, C1Z2);
  C2MaxX := Max (C2X1, C2X2);
  C2MaxY := Max (C2Y1, C2Y2);
  C2MaxZ := Max (C2Z1, C2Z2);

  Write ('POINT ');
  If (PX < C2MinX) or (PY < C2MinY) or (PZ < C2MinZ)
  or (PX > C2MaxX) or (PY > C2MaxY) or (PZ > C2MaxZ) then
    Write ('DOES NOT LIE')
  else
    Write ('LIES');
  Writeln (' INSIDE 2ND CUBE');

  Write ('1ST CUBE ');
  If (C1MinX < C2MinX) or (C1MinY < C2MinY) or (C1MinZ < C2MinZ)
  or (C1MaxX > C2MaxX) or (C1MaxY > C2MaxY) or (C1MaxZ > C2MaxZ)
  then
    Write ('DOES NOT LIE')
  else
    Write ('LIES');
  Writeln (' INSIDE 2ND CUBE');
end.
```

```
{3.4}
program Thr4T88;
{ -- This program produces an alphabetical list of permutations. }
  type
    String6  = Array [1..6] of String[1];
    PermType = Array [1..720] of String[6];
  var
    Number, I: Integer;
    Letters:   String[6];
    S:         String6;
    Perm:      PermType;
    Total:     Integer;

procedure Permute ({Using}     N:     Integer;
                   {Giving} var S:     String6;
                            var Perm:  PermType;
                            var Total: Integer);
{ -- This procedure will interchange the elements in Array S. }
  const
    Empty = '';
  var
    Temp: String[1];
    I, J: Integer;

begin
  If N > 1 then
    begin
      Permute (N - 1, S, Perm, Total);
      for I := N - 1 downto 1 do begin
        {Interchange the elements in S[N] and S[I] }
        Temp := S[N];  S[N] := S[I];  S[I] := Temp;
        Permute (N - 1, S, Perm, Total);
        Temp := S[N];  S[N] := S[I];  S[I] := Temp;
      end;  { -- for I }
    end  { -- if then }
  else
    begin
      Inc(Total);
      Perm[Total] := Empty;
      for J := 1 to Number do
        Perm[Total] := Perm[Total] + S[J];
    end;
end;  {procedure}

procedure Alphabetize (var Perm: Permtype;  Total: Integer);
{ -- This procedure alphabetizes permutations w/insertion sort. }
  var
    I, Index: Integer;
    Temp:     String[6];

begin
  for I := 2 to Total do begin
    Index := I;
    while (Perm[Index] < Perm[Index-1]) and (Index > 1) do begin
      Temp := Perm[Index];
```

```
      Perm[Index] := Perm [Index-1];
      Perm[Index-1] := Temp;
      Dec(Index);
    end;
  end;
end;  { -- procedure }

procedure Display (var Perm: PermType;  Total: Integer);
{ -- This procedure displays the unique permutations in the list.}
  var
    Total2, I: Integer;

begin
  Writeln (Perm[1]);
  Total2 := 1;
  for I := 2 to Total do
    if Perm[I] <> Perm[I-1] then begin
      Writeln (Perm[I]);
      Inc(Total2);
    end;
  Writeln ('TOTAL= ', TOTAL2);
end; { -- procedure }

      { -- Main program }
begin
  Write ('Enter letters: ');  Readln (Letters);
  Number := Length(Letters);
  for I := 1 to Number do
    S[I] := Copy(Letters, I, 1);
  Total := 0;
  Permute (Number, S, Perm, Total);
  Alphabetize (Perm, Total);
  Display (Perm, Total);
end.
```

```
{3.5}
program Thr5T88;
{ -- This program will control the movements of a snake. }
uses Crt;
  const
    SnakeLen = 25;
  var
    V, H, X, Y:     Byte;
    I:              Integer;
    VCoord, HCoord: Array [1..SnakeLen] of Byte;
    FrontHV, EndHV: Byte;
    Ch:             Char;
    InvalidKey:     Boolean;
```

```
begin
  ClrScr;
  InvalidKey := False;
  V := 12;  H := 40 - (SnakeLen div 2);  GotoXY (H,V);
  FrontHV := 0;   EndHV := 1;
  { -- Center snake (asterisks) on the screen }
  for I := H to (H + SnakeLen - 1) do begin
    Write ('*');
    Inc(FrontHV);
    VCoord[FrontHV] := V;
    HCoord[FrontHV] := I;
  end;
  repeat until KeyPressed;
  Ch := ReadKey;

  repeat
    H := HCoord[FrontHV];
    V := VCoord[FrontHV];
    for I := 1 to 2000 do
      If KeyPressed then Ch := ReadKey;

    case Upcase(Ch) of
      'I': Dec(V);
      'M': Inc(V);
      'J': Dec(H);
      'K': Inc(H);
    end;

    for I := 1 to SnakeLen do
      if (H = HCoord[I]) and (V = VCoord[I]) then
        InValidKey := True;

    if InValidKey or (V = 0) or (V = 25) or (H = 0) or (H = 80)
      then InvalidKey := True
    else begin
      GotoXY (H,V);  Write ('*');
      Y := HCoord[EndHV];
      X := VCoord[EndHV];
      GotoXY (Y,X);  Write (' ');
      HCoord[EndHV] := H;
      VCoord[EndHV] := V;
      Inc(FrontHV);
      if FrontHV > SnakeLen then
        FrontHV := 1;
      Inc(EndHV);
      If EndHV > SnakeLen then
        EndHV := 1;
    end; { -- else }
  until InvalidKey;
end.
```

```
{3.6}
program Thr6T88;
{ -- This program will solve two linear equations. }
  type
    String15 = String[15];
  var
    E1, E2, Eq:  String15;
    A1, B1, C1,
    A2, B2, C2:  Integer;
    St, Den,
    NumX, NumY:  Integer;

function Vaal ({Using} Eq: String15;
                    var St: Integer): {Giving} Integer;
{ -- This function determines the coefficient for a term. }
  var
    Md:       String[5];
    En, Sygn: Integer;
    Result:   Integer;
    Coef:     Integer;
begin
  { -- Find Starting position ST of value }
  Sygn := 1;   { -- Default to 1 for positive unsigned #s }
  Md := Copy(Eq, St, 1);
  if Md = '=' then begin
    Inc(St);
    Md := Copy(Eq, St, 1);
  end;
  if Md = 'X' then
    begin
      Vaal := 1;  Inc(St);  Exit;
    end
  else if Md = '+' then
    Inc(St)
  else if Md = '-' then
    begin
      Sygn := -1;  Inc(St);
    end;

  { -- Find ending position EN of value }
  En := St;  Vaal := 0;  Md := Copy(Eq, En, 1);
  while (En <= Length(Eq)) and
        (Md <> 'X') and (Md <> 'Y') and (Md <> '=') do
  begin
    Md := Copy(Eq, En, 1);
    Inc(En);
  end;
  Dec(En);
  if (Md = 'X') or (Md = 'Y') or (Md = '=') then
    Dec(En);
  if Md = '=' then
    Sygn := -Sygn;
  if St > En then begin
    Vaal := Sygn;  Inc(St);  Exit;
  end;
```

```
  { -- Determine value }
  Md   := Copy (Eq, St, En - St + 1);
  Val (Md, Coef, St);
  Vaal := Sygn * Coef;
  St   := En + 2;
end;  { -- function }

{ -- Main routine }
begin
  Write ('Enter equation 1: ');  Readln (E1);
  Write ('Enter equation 2: ');  Readln (E2);
  St := 1;
  A1 := Vaal(E1, St);
  B1 := Vaal(E1, St);
  C1 := Vaal(E1, St);
  St := 1;
  A2 := Vaal(E2, St);
  B2 := Vaal(E2, St);
  C2 := Vaal(E2, St);

  Den  := A1*B2 - A2*B1;
  NumX := C1*B2 - C2*B1;
  NumY := A1*C2 - A2*C1;
  if Den = 0 then
    Writeln ('NO UNIQUE SOLUTION EXISTS.')
  else begin
    Write   ('XSOLUTION= ', NumX / Den : 3:1);
    Writeln ('   YSOLUTION= ', NumY / Den : 3:1);
  end;
end.
```

```
{3.7}
program Thr7T88;
{ -- This program display all semi-perfect #s between 2 and 35. }
uses Crt;
  type
    ArrayType = Array [1..20] of Byte;
  var
    Factors:      ArrayType;
    Num, Di, Max: Byte;
    Combo:        Byte;

procedure PrintCombos ({Using} Factors:  ArrayType;
                                 Combo:  Byte;
                                   Len:  Byte;  Num: Byte);
{ -- This procedure displays Combo character combinations of Len }
  var
    A:            ArrayType;
    N, I, Q, Sum: Byte;
```

```
begin
  for I := 1 to Combo do
    A[I] := Combo - I + 1;
  Dec(A[1]);  N := 1;

  while N <= Combo do begin
    Inc(A[N]);
    if A[N] <= Len - N + 1 then begin
      for I := N - 1 downto 0 do
        A[I] := A[I+1] + 1;

      { -- One combination produced, Now Check for Semi-perfect }
      Sum := 0;
      for I := 1 to Combo do
        Sum := Sum + Factors[ A[I] ];
      if Sum = Num then begin
        Write (Num:2, '      ', Factors[ A[Combo] ]);
        for I := Combo - 1 downto 1 do
          Write (' + ', Factors[ A[I] ]);
        Writeln;
      end;      { -- if Sum }
      N := 0;  { -- Keep N at value 1 }
    end;        { -- if A[N] }
    Inc(N);

  end;  { -- while }
end;  { -- procedure }

begin
  ClrScr;
  Writeln ('SEMI #  EXAMPLE(S)');
  for Num := 2 to 34 do begin
    Max :=0;
    for Di := 1 to (Num Div 2) do
      if (Num mod Di) = 0 then begin
        Inc(Max);
        Factors[Max] := Di;
      end;  { -- If }
    for Combo := 2 to Max do
      PrintCombos (Factors, Combo, Max, Num);
  end;  { -- for Num }
end.
```

```
{3.8}
program Thr8T88;
{ -- This program will keep score for a bowler. }
uses Crt;
  var
    I, J, Fr,
    CommaPos, Len: Byte;
    A:             Array [1..10] of String[3];
    Frames:        String [40];
    Md:            Char;
    Look, Sum:     Array [0..10] of Integer;
    AA:            Array [1..10,1..3] of Integer;

begin
  ClrScr;
  Write ('Enter frames: ');  Readln (Frames);
  Frames := Frames + ',';
  for I := 1 to 10 do begin
    CommaPos := Pos (',', Frames);
    A[I] := Copy(Frames, 1, CommaPos - 1);
    Frames := Copy(Frames,CommaPos + 1,Length(Frames) - CommaPos);
  end;

  Writeln;
  Writeln ('-1- -2- -3- -4- -5- -6- -7- -8- -9- -10-');
  Writeln ('---!---!---!---!---!---!---!---!---!---!');
  for I := 1 to 10 do
    Write (A[I]: 3, '!');
  Writeln;

  { -- Assign values to A FRames according to X, /, or pins }
  for Fr := 1 to 10 do begin
    AA[Fr,2] := 0;
    for J := 1 to Length(A[Fr]) do begin
      Md := A[Fr,J];
      if Md = 'X' then
        begin
          AA[Fr,J] := 10;  Look[Fr] := 2;
        end
      else if Md = '/' then
        begin
          AA[Fr,J] := 10 - AA[Fr,J-1];  Look[Fr] := 1;
        end
      else
        if Md = '-' then
          AA[Fr,J] := 0
        else begin
          AA[Fr,J] := Ord(Md) - Ord('0');  Look[Fr] := 0;
        end;
    end;  { -- for J }
  end;  { -- for Fr }

  { -- Assign Frame values with Look ahead }
  Sum[0] := 0;
  for Fr := 1 to 10 do begin
```

```
    Sum[Fr] := Sum[Fr-1] + AA[Fr,1] + AA[Fr,2];
    if Look[Fr] > 0 then
      if Look[Fr] = 1 then  { -- A spare / needs 1 more value }
        if Fr = 10 then
          Sum[Fr] := Sum[Fr] + AA[Fr,3]
        else
          Sum[Fr] := Sum[Fr] + AA[Fr+1,1]
      else   { -- A strike X needs 2 more values }
        if Fr = 10 then
          Sum[Fr] := Sum[Fr] + AA[Fr,3]
        else
          begin
            Sum[Fr] := Sum[Fr] + AA[Fr+1,1] + AA[Fr+1,2];
            if Fr < 9 then
              if AA[Fr+1,1] = 10 then
                Sum[Fr] := Sum[Fr] + AA[Fr+2,1];
          end;

    Len := Trunc (Ln(Sum[Fr]) / Ln(10)) + 1;
    Write (Sum[Fr]: Len, '': 3 - Len, '!');
  end;  { -- for Fr }
  Writeln;
  for I := 1 to 40 do Write ('-');
  Writeln;
end.
```

```
{3.9}
program Thr9T88;
{ -- This program will convert a real from one base to another. }
  const
    Digits = '0123456789ABCDEF';
  var
    M, N, I, J, MdVal: Byte;
    Num:               String[10];
    MDigits, NDigits:  Byte;
    Md:                Char;
    Sum:               Real;
    NumArray:          Array [0..8] of Byte;

function Power({Using} Base: Real;
                   Exponent: Byte): {Giving} Real;
{ -- This function returns Base^Exponent. }
  var
    I: Integer;
    P: Real;
begin
  P := 1;
  for I := 1 to Exponent do
    P := P * Base;
  Power := P;
end;
```

```
begin
  Write ('Enter M, N, #: ');
  Readln (M, N, Num);
  Write (Copy(Num, 2, 2));
  MDigits := Length(Num) - 3;
  Num := Copy(Num, 4, MDigits);

  NDigits := 1;
  while (Power((1/N),NDigits) > Power((1/M),MDigits))
  and (NDigits < 7) do
    Inc(NDigits);

  { -- SUM = Base 10 # of Num }
  Sum := 0;
  for I := 1 to MDigits do begin
    Md := Num[I];
    MdVal := Ord(Md) - Ord('0');
    if MdVal > 9 then
      MdVal := MdVal - 7;
    Sum := Sum + MdVal / Power(M,I);
  end;

  { -- Convert base 10 decimal to Base N fraction }
  for I := 1 to NDigits + 1 do begin
    Sum := Sum * N;
    NumArray[I] := Trunc(Sum);
    Sum := Sum - NumArray[I];
  end;

  { -- Print fraction with last digit rounded at NDigits + 1 }
  for I := 1 to NDigits - 1 do
    Write (Copy(Digits, NumArray[I] + 1, 1));
  if NumArray[NDigits+1] >= (N / 2) then
    Inc( NumArray[NDigits] );
  Writeln (Copy(Digits, NumArray[NDigits] + 1, 1));
end.
```

```
{3.10}
program Thr10T88;
{ -- This program computes the composition of P (Q) and Q (P) }
  type
    ArrayType = Array [0..5] of Integer;
  var
    POrder, QOrder, I: Integer;
    PCo, QCo:          ArrayType;

procedure ComputeComp ({Using} PCo, QCo: ArrayType;
                          POrder, QOrder: Integer);
{ -- Compute composition of P of Q }
  var
    ProdOrder, CompOrder: Integer;
    I, J, K, L, Ind:      Byte;
    PofQ, Prod, Prod2:    Array [0..25] of Integer;

begin
  CompOrder := POrder * QOrder;
  for I := 0 to CompOrder do
    PofQ[I] := 0;
  for I := 0 to POrder do
    if PCo[I] <> 0 then
      if I = 0 then
        PofQ[0] := PCo[0]
      else begin
        for J := 0 to QOrder do
          Prod[J] := QCo[J];
        ProdOrder := QOrder;
        If I > 1 then
          for Ind := 1 to I-1 do begin
            for J := 0 to ProdOrder + QOrder do
              Prod2[J] := 0;
            for J := 0 to ProdOrder do
              for K := 0 to QOrder do
                Prod2[J+K] := Prod2[J+K] + Prod[J]*QCo[K];
            ProdOrder := J + K;
            for L := 0 to ProdOrder do begin
              Prod[L] := Prod2[L];  Prod2[L] := 0;
            end;  { -- for L }
          end;  { -- for Ind }
          for J := 0 to ProdOrder do
            Prod[J] := Prod[J] * PCo[I];
          for J := ProdOrder downto 0 do
            PofQ[J] := PofQ[J] + Prod[J]
      end;  { -- else begin }

  { -- Print composition }
  for I := CompOrder downto 0 do begin
    if I < CompOrder then
      Write (' + ');
    Write (PofQ[I], 'X**', I);
  end;
  Writeln;
end;
```

```
begin
  Write ('Enter the ORDER of p(x): ');  Readln (POrder);
  for I := POrder downto 0 do begin
    Write ('Enter coefficient for x**',I,': ');  Readln (PCo[I]);
  end;
  Write ('Enter the ORDER of q(x): ');  Readln (QOrder);
  for I := QOrder downto 0 do begin
    Write ('Enter coefficient for x**',I,': ');  Readln (QCo[I]);
  end;

  Write ('P(Q(X))= ');
  ComputeComp (PCo, QCo, POrder, QOrder);
  Write ('Q(P(X))= ');
  ComputeComp (QCo, PCo, QOrder, POrder);
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '89 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T89;
{ -- This program will print an indented phrase on each line. }
uses Crt;
  const
    Phrase = '1989 COMPUTER CONTEST';
  var
    I: Byte;

begin
  ClrScr;
  for I := 1 to 22 do
    Writeln (' ':I, Phrase);
end.


{1.2}
program One2T89;
{ -- This program will translate gigabytes to megabytes. }
  var
    G: Integer;

begin
  Write ('Enter number of gigabytes: ');  { -- less than 30 }
  Readln (G);
  Writeln (G * 1024, ' MEGABYTES');
end.


{1.3}
program One3T89;
{ -- This program displays a word in a backward-L format. }
  var
    Word:   String[15];
    I, Len: Byte;

begin
  Write ('Enter word: ');  Readln (Word);
  Len := Length(Word);
  for I := 1 to Len-1 do
    Writeln (' ':Len-1, Copy(Word, I, 1));
  Writeln (Word);
end.
```

```
{1.4}
program One4T89;
{ -- This program prints a pattern of numbers in pyramid form. }
  var
    N, I: Byte;

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write (' ':10-I, I);
    if I > 1 then
      Write (' ':I*2-3, I);
    Writeln;
  end;
end.


{1.5}
program One5T89;
{ --- This program corrects dates with A.D. or B.C. }
  var
    Era:  String[4];
    Dayt: Integer;

begin
  Write ('Enter date: ');           Readln (Dayt);
  Write ('Enter A.D. or B.C.: ');  Readln (Era);
  if Era = 'A.D.' then
    Writeln (Dayt + 4, ' ', Era)
  else if Dayt > 4 then
    Writeln (Dayt - 4, ' ', Era)
  else
    Writeln (5 - Dayt, ' A.D.');
end.
```

```
{1.6}
program One6T89;
{ -- This program will allow a user access with a password. }
  const
    Pass = 'ITSME';
  var
    PassW: String[10];
    I:     Byte;

begin
  Write ('ENTER PASSWORD: ');  Readln (PassW);
  I := 0;
  while (PassW <> Pass) and (I < 2) do begin
    Writeln ('INVALID PASSWORD');
    Write ('ENTER PASSWORD: ');  Readln (PassW);
    Inc(I);
  end;
  if PassW = Pass then
    Writeln ('YOU HAVE ACCESS')
  else
    Writeln ('YOU ARE TRESPASSING');
end.


{1.7}
program One7T89;
{ -- This program will display the best DBMS. }
  var
    N, Max, I, C, E: Byte;
    Name, Best:      String[9];

begin
  Write ('Enter N: ');  Readln (N);  Max := 0;
  for I := 1 to N do begin
    Write ('Enter DBMS name: ');  Readln (Name);
    Write ('Enter convenience, efficiency: ');   Readln (C, E);
    if C + E > Max then begin
       Max := C + E;  Best := Name;
    end;
  end;
  Writeln (Best, ' IS BEST');
end.
```

```
{1.8}
program One8T89;
{ -- This program displays the unique elements of a list. }
  var
    N:      Integer;
    Num, I: Byte;
    List:   Array[1..10] of Integer;

begin
  Write ('Enter #: ');  Readln (N);
  Num := 0;
  while N <> -999 do begin
    I := 1;
    while (I <= Num) and (N <> List[I]) do Inc(I);
    if I > Num then begin
      Num := I;  List[Num] := N;
    end;
    Write ('Enter #: ');  Readln (N);
  end;
  for I := 1 to Num do Write (List[I], ' ');
end.


{1.9}
program One9T89;
{ -- This program determines how many feet deep of dollar coins
  -- over Texas is equivalent to a given probability. }
  const
    TexasArea = 262134.0;
  var
    DolVol, TexasVol, Prob, InchDeep: Real;

begin
  Write ('Enter probability: ');  Readln (Prob);
  DolVol := (1.5) * (1.5) * (3/32);  { -- Volume Dollar takes }
  TexasVol := TexasArea * (5280.0 * 12.0 * 5280.0 * 12.0);
  InchDeep := (Prob / (TexasVol / DolVol));
  Writeln (InchDeep / 12 :5:0, ' FEET DEEP');
end.
```

```
{1.10}
program One10T89;
{ -- This program will map a logical address to the physical. }
  const
    Base: Array[0..4] of Integer = (219, 2300, 90, 1327, 1952);
    Len:  Array[0..4] of Integer = (600, 14, 100, 580, 96);
  var
    Adr, Seg: Integer;

begin
  Write ('Enter Seg#, Address: ');  Readln (Seg, Adr);
  while Seg <= 4 do begin
    if Adr > Len[Seg] then
      Writeln ('ADDRESSING ERROR')
    else
      Writeln (Base[Seg] + Adr);
    Write ('Enter Seg#, Address: ');  Readln (Seg, Adr);
  end;
end.
```

```
{2.1}
program Two1T89;
{ -- This program prints F(x) for a recursive function given x. }
  var
    F:    Array [1..11] of Integer;
    I, X: Byte;

begin
  Write ('Enter x: ');  Readln (X);
  F[1] := 1;  F[2] := 1;  F[3] := 1;
  I := 3;
  while I < X do begin
    F[I+1] := (F[I] * F[I-1] + 2) div F[I-2];
    Inc(I);
  end;
  Writeln ('F(', X, ')=', F[X]);
end.


{2.2}
program Two2T89;
{ -- This program will print the prime factors of a number. }
  var
    I, Num: Integer;

begin
  Write ('Enter #: ');  Readln (Num);
  while Num > 1 do begin
    I := 2;
    while (Num mod I) > 0 do
      Inc(I);
    Write (I);
    Num := Num div I;
    if Num > 1 then Write (' X ');
  end;
end.
```

```
{2.3}
program Two3T89;
{ -- This program will display a word without its vowels. }
  const
    Vow = 'AEIOU';
  var
    Word: String[15];
    Ch:   Char;
    I:    Byte;

begin
  Write ('Enter word: ');  Readln (Word);
  for I := 1 to Length(Word) do begin
    Ch := Word[I];
    if Pos(Ch, Vow) = 0 then
      Write (Ch);
  end;
end.


{2.4}
program Two4T89;
{ -- This program produces the shortest possible identifiers. }
  var
    I, J, K: Byte;
    A:       Array[1..6] of String[10];
    S:       String[10];

begin
  for I := 1 to 6 do begin
    Write ('Enter name: ');  Readln (A[I]);
  end;
  for I := 1 to 6 do begin
    K := 1;  S := Copy(A[I], 1, 1);
    for J := 1 to 6 do
      { -- If S is same as beginning of another var, add letter. }
      while (I <> J)
      and (S = Copy(A[J], 1, K))
      and (K < Length(A[I])) do begin
        Inc(K);  S := S + Copy(A[I], K, 1);
      end;
    Writeln (S);
  end;  { -- for I }
end.
```

```
{2.5}
program Two5T89;
{ -- This program prints the # of distinguishable permutations. }
  var
    Word:   String[15];
    I, Len: Byte;
    LetPos: Integer;
    Let:    Array[1..26] of Byte;
    Num:    LongInt;

begin
  Write ('Enter word: ');  Readln (Word);
  Len := Length(Word);
  for I := 1 to 26 do Let[I] := 0;

  { -- Calculate Len factorial (assuming all different letters) }
  Num := 1;
  for I := 1 to Len do Num := Num * I;

  { -- Divide out of Num the factorials of the same letters }
  for I := 1 to Len do begin
    LetPos := Ord(Word[I]) - 64;
    Let[LetPos] := Let[LetPos] + 1;
    if Let[LetPos] > 1 then
      Num := Num div Let[LetPos];
  end;

  Writeln (Num);
end.
```

```
{2.6}
Program Two6T89;
{ -- This program underlines parts of a sentence between 2 *'s. }
uses Crt;
  const
    Dash = '-';
  var
    Sent:   String[40];
    I, Col: Byte;
    Under:  Boolean;
    Ch:     String[1];

begin
  Write ('Enter Sentence: ');  Readln (Sent);
  ClrScr;  Writeln (Sent);

  Under := False;  Col := 0;
  for I := 1 to Length(Sent) do begin
    Ch := Copy(Sent, I, 1);
    if Ch = '*' then
      { -- Change to Underline mode or un-underline mode. }
      Under := not Under
    else  { -- Display Char and underline if in underline mode. }
      begin
        Inc(Col);
        GotoXY (Col, 3);  Write (Ch);
        if Under then begin
          GotoXY (Col, 4);  Write (Dash);
        end;
      end;
  end;
  Writeln;
end.
```

```
{2.7}
program Two7T89;
{ -- This program will compute an expression containing + - * / }
  var
    St:                    String[10];
    NumSt:                 String[4];
    Num1, Num2, I, Result: Integer;
    Ch, Symbol:            Char;

begin
  Write ('Enter expression: ');  Readln (St);  NumSt := '';
  { -- Parse first number in Num1 and second number in Num2 }
  for I := 1 to Length(St) do begin
    Ch := St[I];
    if Ch in ['+', '-', '*', '/'] then
      begin
        Symbol := Ch;
        Val(NumSt, Num1, Result);  NumSt := '';
      end
    else
      NumSt := NumSt + Ch;
  end;
  Val (NumSt, Num2, Result);

  Case Symbol of
    '+':  Writeln (Num1 + Num2);
    '-':  Writeln (Num1 - Num2);
    '*':  Writeln (Num1 * Num2);
    '/':  Writeln (Num1 div Num2);
  end;
end.
```

```
{2.8}
program Two8T89;
{ -- This program will display the saddle point of a matrix. }
  var
    Rows, Cols, I, J, K:     Byte;
    Mat: Array[1..5,1..5] of Integer;
    Small, Large:            Boolean;

begin
  Write ('Enter # Rows, # Cols: ');  Readln (Rows, Cols);
  for I := 1 to Rows do
    for J := 1 to Cols do begin
      Write ('Enter Row', I, ' Col', J, ': ');
      Readln (Mat[I,J]);
    end;

  for I := 1 to Rows do
    for J := 1 to Cols do begin
      Small := True;
      for K := 1 to Cols do
        if (K <> J) and (Mat[I,J] >= Mat[I,K]) then
          Small := False;
      if Small then begin
        Large := True;
        for K := 1 to Rows do
          if (K <> I) and (Mat[I,J] <= Mat[K,J]) then
            Large := False;
        if Large then begin
          Write ('SADDLE POINT = ');
          Writeln (Mat[I,J], ' AT ROW ', I, ' COL ', J);
        end;
      end;  { -- if Small }
    end;  { -- for J }
end.
```

```
{2.9}
program Two9T89;
{ -- This program will sort a set of dates in increasing order. }
  const
    Mo: Array[1..12] of String[9] = ('JANUARY', 'FEBRUARY',
      'MARCH', 'APRIL', 'MAY', 'JUNE', 'JULY', 'AUGUST',
      'SEPTEMBER', 'OCTOBER', 'NOVEMBER', 'DECEMBER');
  var
    I, J, N, Temp: Integer;
    M:     Array[1..10] of String[9];
    D, Y:  Array[1..10] of Integer;
    Sort:  Array[1..10] of LongInt;
    Index: Array[1..10] of Integer;

begin
  Write ('Enter # of dates: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter month: ');  Readln (M[I]);
    Write ('Enter day:   ');  Readln (D[I]);
    Write ('Enter year:  ');  Readln (Y[I]);
    Writeln;

    { -- Combine Year, Month, Day (in that order) for sorting. }
    J := 1;
    while (J < 13) and (M[I] <> Mo[J]) do Inc(J);
    Sort[I] := ((Y[I] * 100) + J) * 100 + D[I];
    Index[I] := I;
  end;

  { -- Sort dates according to values in Sort[] and swap Index. }
  for I := 1 to N - 1 do
    for J := I + 1 to N do
      if Sort[Index[I]] > Sort[Index[J]] then begin
        Temp := Index[I]; Index[I] := Index[J]; Index[J] := Temp;
      end;

  for I := 1 to N do
    Writeln (M[Index[I]], ' ', D[Index[I]], ' ', Y[Index[I]]);
end.


{2.10}
program Two10T89;
{ -- This program displays class grades and the averages. }
uses Crt;
  const
    Name: Array[1..5] of String[8] =
     ('D. WOOLY', 'M. SMITH', 'C. BROWN', 'R. GREEN', 'T. STONE');
    Quiz: Array[1..5,1..4] of Byte =
     ((100, 92, 90, 90),  (55, 75, 70, 65),  (94, 70, 62, 70),
       (90, 74, 80, 85),  (85, 98, 100, 70));
  var
    I, J, Scr:  Byte;
    Sum, Total: Real;
```

```
begin
  for Scr := 1 to 2 do begin
    ClrScr;
    if Scr = 2 then begin
      Writeln ('           MS. HEINDEL''S MUSIC CLASS');
      Writeln ('                  FINAL GRADES');
      Writeln ('                  SPRING 1989');
      Writeln;
    end;
    Write ('  NAME       Q1     Q2     Q3     Q4');
    if Scr = 2 then
      Writeln ('    AVERAGE')
    else
      Writeln;
    Writeln;

    for I := 1 to 5 do begin
      Write (Name[I]);  Sum := 0;
      for J := 1 to 4 do begin
        Write (Quiz[I,J]:7);  Sum := Sum + Quiz[I,J];
      end;
      if Scr = 2 then
        Writeln (' ':4, Sum / 4: 5:2)
      else
        Writeln;
    end;
    Writeln;
    if Scr = 1 then begin
     Write  ('Enter 5 grades for quiz 4: ');
     Readln(Quiz[1,4], Quiz[2,4], Quiz[3,4], Quiz[4,4],Quiz[5,4]);
    end;
  end;  { -- for Scr }

  { -- Display Column averages and Class average. }
  Write ('AVERAGE:');  Total := 0;
  for I := 1 to 4 do begin
    Sum := 0;
    for J := 1 to 5 do
      Sum := Sum + Quiz[J,I];
    Write ('  ', Sum / 5: 5:2);
    Total := Total + Sum;
  end;
  Writeln;  Writeln;
  Writeln ('CLASS AVERAGE: ', Total / 20: 5:2);
end.
```

```
{3.1}
program Thr1T89;
{ -- This program will determine if a word is correctly spelled. }
  var
    St, Part: String[12];
    Correct:  Boolean;
    I, Len:   Byte;

begin
  Write ('Enter word: ');  Readln (St);
  Len := Length(St);   Correct := True;

  { -- Check for E before suffixes ING, IBLE, ABLE }
  if Len >= 4 then begin
    Part := Copy(St, Len-2, 3);
    if (Part = 'ING') and (Copy(St, Len-3, 1) = 'E') then
      Correct := False;
  end;
  if Len >= 5 then begin
    Part := Copy(St, Len-3, 4);
    if ((Part = 'IBLE') or (Part = 'ABLE')) and
    (Copy(St, Len-4, 1) = 'E')
      then Correct := False;
  end;

  { -- Check if IE after C. }
  Part := St;  I := Pos('IE', Part);
  while (I > 0) and Correct do begin
    Dec(I);
    if I >= 1 then
      if Copy(Part, I, 1) = 'C' then Correct := False;
    Part := Copy (Part, I+3, Length(Part) - (I+2));
    I := Pos('IE', Part);
  end;

  { -- Check if EI not after C. }
  Part := St;  I := Pos('EI', Part);
  while (I > 0) and Correct do begin
    Correct := False;
    if I >= 2 then
      if Copy(Part, I-1, 1) = 'C' then Correct := True;
    Part := Copy (Part, I+3, Length(Part) - (I+2));
    I := Pos('EI', Part);
  end;

  { -- Check for 3 consecutive same letters. }
  I := 1;
  while (I <= Len-2) and Correct do begin
    if (Copy(St,I,1) = Copy(St,I+1,1))
    and (Copy(St,I,1) = Copy(St,I+2,1))
      then Correct := False;
    Inc(I);
  end;
  if Correct then
    Writeln ('CORRECT')
```

```
  else
    Writeln ('MISSPELLED');
end.



{3.2}
program Thr2T89;
{ -- This program finds the positive root of V for an equation. }
  const
    P: Array[1..5] of Real = (0.05, 0.7, 10.0, 70.0, 30.0);
  var
    I:              Byte;
    ZeroFound:      Boolean;
    Neg, Pos, T,
    V, VTry, NextV: Real;

function FNA(V: Real): Real;
{ -- This function computes the value of P for the equation. }
begin
  FNA := P[I]*V*V*V*14.14 - P[I]*V*9062.599 - 23511.9*V*V +
         988686.1*V - 400943.0;
end;

begin
  for I := 1 TO 5 do begin
    if I = 5 then begin  { -- Allow for 1 input value }
       Writeln;
       Write ('Enter value for P: ');  Readln (P[5]);
    end;
    VTry := 0;  ZeroFound := False;
    repeat
      NextV := VTry + 1;
      if (FNA(VTry) * FNA(NextV) <= 0) and (FNA(NextV) <> 0) then
      { -- Sign change has occurred }
      begin
        Neg := VTry;  Pos := NextV;
        if FNA(Neg) > FNA(Pos) then begin
          T := Neg;  Neg := Pos;  Pos := T;
        end;
        repeat
          V := (Neg + Pos) / 2.0;
          if FNA(V) < 0 then Neg := V else Pos := V;
        until ABS(Neg - Pos) <= 0.00005;
        Writeln ('P = ', P[I]:5:2, '  V = ', V:6:4);
        ZeroFound := True;
      end;
      VTry := VTry + 1;
    until ZeroFound or (VTry > 2);
  end;  { -- next I }
end.
```

```
{3.3}
program Thr3T89;
{ -- This program will magnify an input positive integer. }
uses Crt;
  const
    Num: Array[0..9] of String[7] = ('123567', '36', '13457',
         '13467', '2346', '12467', '124567', '136',
         '1234567', '12346');
  var
    NSt: String[4];
    Col, Part, I, J, K, N, Magn: Integer;

procedure DisplayPart (Part: Integer);
{ -- This procedure displays a vertical or horizontal line seg. }
begin
  Case Part of
    1: begin
         GotoXY (Col, 1);  for K := 1 to Magn do Write ('****');
         Writeln;
       end;
    2: begin
         for K := 1 to Magn*2+1 do begin
           GotoXY(Col, K);  Write('*');  end;
       end;
    3: begin
         for K := 1 to Magn*2+1 do begin
           GotoXY(Col+Magn*4-1, K);  Write('*');  end;
       end;
    4: begin
         GotoXY(Col, Magn*2+1);
         for K := 1 to Magn do Write ('****');  Writeln;
       end;
    5: begin
         for K := Magn*2+1 to Magn*4+1 do begin
           GotoXY(Col, K);  Write('*');  end;
       end;
    6: begin
         for K := Magn*2+1 to Magn*4+1 do begin
           GotoXY(Col+Magn*4-1, K);  Write('*');  end;
       end;
    7: begin
        GotoXY(Col, Magn*4+1);
        for K := 1 to Magn do Write ('****');  Writeln;
      end;
  end;
end;

begin
  Write ('Enter number: ');  Readln (NSt);
  Write ('Enter magnification: ');  Readln (Magn);
  ClrScr;
  for I := 1 to Length(NSt) do begin
    N := Ord(NSt[I]) - 48;
    Col := (I-1) * Magn * 6 + 1;
    for J := 1 to Length(Num[N]) do begin
```

```
      Part := Ord(Num[N,J]) - 48;
      DisplayPart(Part);
    end;
  end;
end.


{3.4}
program Thr4T89;
{ -- This program produces a calendar for a given month/year. }
{ -- January 1, 1901 is a Tuesday. }
uses Crt;
  const
    Mo: Array[1..12] of String[9] = ('JANUARY', 'FEBRUARY',
      'MARCH', 'APRIL', 'MAY', 'JUNE', 'JULY', 'AUGUST',
      'SEPTEMBER', 'OCTOBER', 'NOVEMBER', 'DECEMBER');
    DaysInMo: Array[1..12] of Byte =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
  var
    Year, Days:                    Integer;
    Month, Day, Col, Leap, I, Mid: Byte;

begin
  Write ('Enter month, year: ');  Readln (Month, Year);
  ClrScr;
  Mid := 2 + (26 - (Length(Mo[Month]) + 5)) div 2;
  Writeln (' ':Mid, Mo[Month], ' ', Year);
  Writeln ('   S   M   T   W   T   F   S');
  Writeln ('  --------------------------');

  { -- # of days from 1/1/1901 to the last day of prior month. }
  Days := (Year - 1901) * 365 + ((Year - 1901) div 4);
  for I := 1 to Month - 1 do
    Days := Days + DaysInMo[I];
  if (Month > 2) and (Year mod 4 = 0) then
    Inc(Days);

  { -- Determine first day of month. }
  Day := (Days + 1) mod 7;
  { -- Day =0 (Mon), =1 (Tue), =5 (Sat), =6 (Sun) }
  Col := (Day + 1) mod 7;
  { -- Day = 0,1,2,3,4,5,6 Sun,Mon,Tue..Sat }
  Leap := 0;
  if (Month = 2) and (Year mod 4 = 0) then  { -- Leap year month }
    Leap := 1;

  { -- Display Month Calendar }
  if Col > 0 then Write (' ':Col*4);
  for I := 1 to DaysInMo[Month] + Leap do begin
    Write (I:4);
    Col := (Col + 1) mod 7;
    if Col = 0 then Writeln;
  end;
end.
```

```
{3.5}
program Thr5T89;
{ -- This program positions 5 queens on the board so none attack.}
uses Crt;
  const
    Dimen = 5;
  type
    Board = Array [1..8] of Byte;
  var
    I, Row, Col:   Byte;
    Configuration: Board;

function IsSafe (Configuration: Board; Row, Col: Byte): Boolean;
{ -- This function returns True if no queen can attack another. }
  var
    I:      Byte;
    Safety: Boolean;
  begin
    Safety := True;
    for I := 1 to Col-1 do
      if ((Configuration[I] + I) = (Row + Col))
      or ((Configuration[I] - I) = (Row - Col))
      or  (Configuration[I] = Row) then
        Safety := False;
      IsSafe := Safety
  end;

begin
  ClrScr;
  Writeln ('ROWS = 1 2 3 4 5');
  Writeln ('----------------');
  Writeln ('COLUMNS');
  Col := 1;  Row := 1;
  repeat
    while (Row <= Dimen) and (Col <= Dimen) do
      if IsSafe(Configuration, Row, Col) then
        { -- Advance the Column }
        begin
          Configuration[Col] := Row;  Inc(Col);  Row := 1
        end
      else
        Inc(Row);

    if (Row = Dimen + 1) then begin  { -- Retreat the Column }
      Dec(Col);
      Row := Configuration[Col] + 1
    end;

    if (Col = Dimen + 1) then begin
      { -- Display Solution and retreat column. }
      Write (' ':7);
      for I := 1 to Dimen do
        Write (Configuration[I], ' ');
      Writeln;
      Dec(Col);
```

```
      Row := Configuration[Col] + 1
    end;
  until (Col = 1) and (Row = Dimen + 1);
end.
```

```
{3.6}
program Thr6T89;
{ -- This program prints the product of 2 large integers in Base.}
  var
    AStr, BStr:           String[31];
    LenA, LenB:           Byte;
    A, B, Prod:           Array[1..61] of Byte;
    I, J, S, Carry, Base: Byte;
    Sign:                 -1..1;

begin
  Write ('Enter base: ');  Readln (Base);
  Write ('Enter first integer:  ');  Readln (AStr);
  Write ('Enter second integer: ');  Readln (BStr);

  { -- Determine if signs are positive or negative, display sign.}
  Sign := 1;
  if Copy(AStr, 1, 1) = '-' then begin
    AStr := Copy(AStr, 2, Length(AStr)-1);   Sign := -1;
  end;
  if Copy(BStr, 1, 1) = '-' then begin
    BStr := Copy(BStr, 2, Length(BStr)-1);  Sign := Sign * -1;
  end;
  if Sign < 0 then Write ('-');

  { -- Store String digits into numerical arrays. }
  LenA := Length(AStr);  LenB := Length(BStr);
  for I := LenA downto 1 do
    A[LenA - I + 1] := Ord(AStr[I]) - 48;
  for I := LenB downto 1 do
    B[LenB - I + 1] := Ord(BStr[I]) - 48;
  for I := 1 to 61 do Prod[I] := 0;

  { -- Multiply 2 numbers as a person would with carries. }
  for I := 1 to LenB do begin
    Carry := 0;
    for J := 1 to LenA do begin
      S := I + J - 1;
      Prod[S] := Prod[S] + B[I] * A[J] + Carry;
      Carry := Prod[S] div Base;;
      Prod[S] := Prod[S] - Carry * Base;
    end;
    If Carry > 0 then Prod[S+1] := Carry;
  end;

  { -- Display product }
  if Carry > 0 then Write (Prod[S+1]);
  for I := S downto 1 do
    Write (Prod[I]);

end.
```

```
{3.7}
program Thr7T89;
{ -- This program computes most efficient change without a coin. }
  const
    Coin: Array[1..4] of String[7] =
       ('QUARTER', 'DIME', 'NICKEL', 'PENNY');
    CVal: Array[1..4] of Byte = (25, 10, 5, 1);
  var
    CoinM:         String[7];
    Num:           Array[1..4] of Byte;
    Cost, Amount: Real;
    Change, I, C: Byte;

procedure MakeChange (X, St, En: Integer);
{ -- Gives most efficient change of X using CoinValues[St..En] }
  var I: Integer;
begin
  for I := St to En do begin
    Num[I] := X div CVal[I];
    X := X - Num[I] * CVal[I];
  end;
end;

procedure DoMissingCoin (C: Byte);
{ -- Make up change for missing coin (if it was used in solution)}
begin
  if C = 1 then  { -- NO Quarters }
    { -- Determine most efficient way withoug quarters }
    MakeChange (Change, 2, 4)
  else if C = 2 then  { -- NO Dimes }
    { -- Add 2 nickels for every dime }
    Num[3] := Num[3] + Num[2] * 2
  else if C = 3 then  { -- NO Nickels }
    { -- IF a nickel then IF at least 1 quarter then
                              Make 3 dimes and 1 less quarter
                            ELSE make 5 more pennies with 1 nickel }
   if Num[3] = 1 then
    if Num[1] > 0 then begin
      Num[2] := Num[2] + 3;  Num[1] := Num[1] - 1; end
    else
      Num[4] := Num[4] + 5;
end;

begin
  Write ('Enter cost, amount: ');  Readln (Cost, Amount);
  Write ('Enter missing coin: ');  Readln (CoinM);
  Change := Trunc((Amount - Cost) * 100 + 0.01);
  MakeChange (Change, 1, 4);  { -- Calculate denominations }

  C := 1;
  while (C < 5) and (CoinM <> Coin[C]) do  Inc(C);
  DoMissingCoin(C);

  { -- Display number of coins of each coin that was used. }
  for I := 4 downto 1 do begin
```

```
    if I <> C then begin
      Write (Num[I],'  ');
      if (I = 4) and (Num[I] <> 1) then Writeln ('PENNIES')
      else begin
        Write (Coin[I]);
        if Num[I] <> 1 then Write ('S');
        Writeln;
      end;
    end;
  end;
  Write ('TOTAL CHANGE RETURNED = ',Change,' CENT');
  if Change <> 1 then Write ('S');
  Writeln;
end.
```

```
{3.8}
program Thr8T89;
{ -- This program displays the coordinates of binary rectangles. }
  var
    A:                              Array [1..6,1..7] of 0..1;
    I, J, K, Num, Den:              Byte;
    RowLen, ColLen, RowSt, ColSt: Byte;
    Rect:                           Boolean;

begin
  { -- Convert 6 numbers to binary representation. }
  for I := 1 to 6 do begin
    Write ('Enter number: ');  Readln (Num);
    Den := 128;
    for J := 6 downto 0 do begin
      Den := Den div 2;  { -- Den = 2^J }
      A[I,7-J] := Num div Den;
      Num := Num - A[I,7-J] * Den;
    end;
  end;
  Writeln;

  { -- Display the 6 row X 7 col grid of 0s and 1s. }
  for I := 1 to 6 do begin
    for J := 1 to 7 do
      Write (A[I,J]);
    Writeln;
  end;
  Writeln;

  { -- Find largest solid rectangles of 1s. }
  for RowLen := 6 downto 2 do
    for ColLen := 7 downto 2 do
      for RowSt := 1 to 7 - RowLen do
        for ColSt := 1 to 8 - ColLen do begin
          Rect := True;
          for I := RowSt to RowSt + RowLen - 1 do begin
            J := ColSt;
            while (J <= ColSt + ColLen - 1) and Rect do begin
              if A[I,J] = 0 then Rect := False;
              J := J + 1;
            end;
          end;  { -- for I }
          if Rect then begin  { -- Display rectangle coordinates }
            Write ('(', RowSt, ',', ColSt, ')');
            Write ('(', RowSt + RowLen - 1, ',');
            Writeln (ColSt + ColLen - 1, ')');
            { -- Clear rectangle 1s to 0s }
            for I := RowSt to RowSt + RowLen - 1 do
              for J := ColSt to ColSt + ColLen - 1 do
                A[I,J] := 0;
          end;
        end;  { -- for ColSt }
end.
```

```
{3.9}
program Thr9T89;
{ -- This program determines the 5 word combination for BINGO. }
  type
    String5 =    String[5];
    OfWord=      Array [1..5] of String[1];
    Array3=      Array [1..5,1..2] of Byte;
    ArrayWord3 = Array [1..5,1..2] of String5;

  const
    LetterValue: Array[1..26] of Byte =
    (9, 14, 1, 16, 20, 5, 10, 2, 21, 17, 6, 25,
    12, 3, 22, 18, 24, 7, 13, 26, 15, 11, 19, 4, 23, 8);

  var
    I, J, K, Sum, Col, Row, MaxCol, St, En,
    WordNum:  Byte;
    Max:      Integer;
    Word:     String5;
    Letter:   Char;
    Letters:  OfWord;
    Highest:  Array3;
    HighWord: ArrayWord3;
    MaxSum:   Array [1..2] of Integer;

procedure UseWord (Word: String5;  Sum: Integer);
{ -- This procedure replaces a word if the sum of new word is >. }
  const
    Bingo = 'BINGO';
  var
    Row, Col: Byte;

begin
  for Col := 1 to 2 do
    for Row := 1 to 5 do
      if Letters[Col] = Copy(Bingo,Row,1) then
        if Sum > Highest [Row, Col] then begin
          Highest  [Row, Col] := Sum;
          HighWord [Row, Col] := Word;
        end;
end;

procedure DisplayValues;
{ -- This procedure displays the two columns of values on screen.}
begin
  Writeln;
  Max := 0;
  for I := 1 to 2 do
    MaxSum[I] := 0;
  St := 1;  En :=2;
  for Row := 1 to 5 do begin
    for Col := St to En do begin
      Write(HighWord [Row, Col]:5, Highest[Row, Col]:4, ' ':3);
      MaxSum[Col] := MaxSum[Col] + Highest[Row, Col];
    end; {for Row}
```

```
    Writeln;
  end; {for Col}

  { -- Determine maximum column and display *** }
  For Col := St to En do begin
    Write (' ': 3 + Col * 3, MaxSum[Col] :3);
    If MaxSum[Col] > Max then begin
      Max := MaxSum[Col];
      MaxCol := Col;
    end;
  end; {for Col}
  Writeln;
  if MaxCol = 1 then
    Writeln (' ':6, '***')
  else
    Writeln (' ':18, '***');
  Writeln;
end;

begin
  HighWord[1,1] := 'BIBLE';  HighWord[1,2] := 'OBESE';
  HighWord[2,1] := 'IDYLL';  HighWord[2,2] := 'TITHE';
  HighWord[3,1] := 'NOISE';  HighWord[3,2] := 'INLET';
  HighWord[4,1] := 'GULLY';  HighWord[4,2] := 'IGLOO';
  HighWord[5,1] := 'OBESE';  HighWord[5,2] := 'TOWER';

  { -- Determine numerical values for given words. }
  for Col := 1 to 2 do
    for Row := 1 to 5 do begin
      Sum := 0;
      for I := 1 to 5 do begin
        Word := HighWord[Row,Col];
        Letter := Word[I];
        Sum := Sum + LetterValue[Ord(Letter) - 64];
      end; {for I}
      Highest[Row,Col] := Sum;
    end;
  repeat
    DisplayValues;
    { -- Allow new words to be entered and computed. }
    Write ('Enter word: ');  Readln (Word);
    while Length(Word) = 5 do begin
      Sum := 0;
      for I := 1 to 5 do begin
        Letter := Word[I];
        Letters[I] := Letter;
        Sum := Sum + LetterValue[Ord(Letter) - 64];
      end; {for I}
      UseWord (Word, Sum);
      Write ('Enter word: ');  Readln (Word);
    end; {while}
  until Word = 'QUIT';
end.
```

```
{3.10}
program Thr10T89;
{ -- This program displays the number of distinguishable
  -- permutations for a cube w/sides input as color symbols. }

  const
    Side: Array[1..6] of String[6] =
    ('TOP', 'FRONT', 'BOTTOM', 'BACK', 'RIGHT', 'LEFT');
  type
    CubeArray = Array[1..6] of Char;
  var
    I, J, K,
    Rot, Num: Byte;
    Cube, C:  CubeArray;
    Unique:   Array[1..24, 1..6] of Char;
    Valid:    Boolean;

procedure Permute (var C: CubeArray;  Rot: Byte);
{ -- Swaps the colors on the squares of the Cube. }
  var
    Temp:   Char;
    Square: Byte;

begin
  if Rot mod 4 > 0 then
    { -- Rotate cube clock-wise about vertical axis }
    begin
      Temp := C[ 2];
      C[2] := C[5];  C[5] := C[4];
      C[4] := C[6];  C[6] := Temp;
    end
  else
  { -- Place a new square ((Rot div 4) + 1) on the top position. }
  begin
    Square := (Rot div 4) + 1;
    C[1] := Cube[Square];
    Case Square of
      1: begin
          for I := 2 to 6 do
            C[I] := Cube[I]
         end;
      2: begin
           C[2] := Cube[3];  C[3] := Cube[4];
           C[4] := Cube[1];  C[5] := Cube[5];  C[6] := Cube[6];
         end;
      3: begin
           C[2] := Cube[4];  C[3] := Cube[1];
           C[4] := Cube[2];  C[5] := Cube[5];  C[6] := Cube[6];
         end;
      4: begin
           C[2] := Cube[1];  C[3] := Cube[2];
           C[4] := Cube[3];  C[5] := Cube[5];  C[6] := Cube[6];
         end;
      5: begin
           C[2] := Cube[2];  C[3] := Cube[6];
```

```
            C[4] := Cube[4];  C[5] := Cube[3];  C[6] := Cube[1];
          end;
        6: begin
            C[2] := Cube[2];  C[3] := Cube[5];
            C[4] := Cube[4];  C[5] := Cube[1];  C[6] := Cube[3];
          end;
    end; { -- case }
  end; { -- if }
end;  { -- Procedure }


begin
  { -- Assign colors to original 4 cubes. }
  { -- [.,#] # is  1= Top, 2= Front, 3= Bot, 4= Bk, 5= Rt, 6= Lt }
  for I := 1 to 6 do begin
    Write ('Enter ', Side[I], ' side: ');  Readln (Cube[I]);
  end;
  Num := 0;

  { -- Rotate cubes and check if it is unique. }
  for Rot := 0 to 23 do begin
    Permute (C, Rot);
    if Rot = 0 then
      Valid := True
    else
      begin
      { -- Check if permuted cube is identical to previous cubes.}
        J := 1;
        repeat
          Valid := False;
          for K := 1 to 6 do
            If C[K] <> Unique[J,K] then Valid := True;
          Inc(J);
        until (J > Num) or not Valid;
      end;

    If Valid then begin  { -- Add new cube to unique cube list }
      Inc(Num);
      for I := 1 to 6 do
        Unique[Num, I] := C[I];
    end;
  end; { Rot }
  Writeln ('NUMBER OF DISTINGUISHABLE CUBES = ', Num);
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '90 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T90;
{ -- This program will display the initials NCNB. }
begin
  Writeln ('NN    N  CCCCC  NN    N  BBBB');
  Writeln ('N N   N  C      N N   N  B   B');
  Writeln ('N  N  N  C      N  N  N  BBBBB');
  Writeln ('N   N N  C      N   N N  B   B');
  Writeln ('N    NN  CCCCC  N    NN  BBBB');
end.


{1.2}
{ -- This program will print the name of the SYSTEM. }
  var
    N: Byte;

begin
  Write ('Enter #: ');  Readln (N);
  Writeln ('SYSTEM ', N);
end.


{1.3}
program One3T90;
{ -- This program will display value of programmers. }
  var
    N: Integer;

begin
  Write ('Enter N: ');  Readln (N);
  Writeln (66 + N, ' BILLION DOLLARS');
end.

{1.4}
program One4T90;
{ -- This program will indicate county for zip code. }
  var
    Zip: String[5];

begin
  Write ('Enter zip code: ');  Readln (Zip);
  if (Zip = '33701') or (Zip = '34685') or (Zip = '34646') then
    Writeln ('PINELLAS')
  else
  if (Zip = '33525') or (Zip = '34249') or (Zip = '34690') then
    Writeln ('PASCO')
  else
    Writeln ('HILLSBOROUGH');
end.
```

```
{1.5}
program One5T90;
{ -- This program will display Hugh McColl's goals. }
  var
    MMM, YYYY: Integer;

begin
  Write ('Enter MMM: ');  Readln (MMM);
  Write ('Enter YYYY: '); Readln (YYYY);
  Writeln ('HUGH MCCOLL WOULD LIKE NCNB TO GROW');
  Writeln ('TO ', MMM, ' BILLION DOLLARS IN ASSETS BY');
  Writeln ('THE YEAR ', YYYY);
end.

{1.6}
program One6T90;
{ -- This program will calculate maximum number of coupons. }
  var
    N, C: LongInt;

begin
  Write ('Enter N associates: ');  Readln (N);
  Write ('Enter C coupons: ');     Readln (C);
  Writeln (Trunc (C / N + 0.999));
end.


{1.7}
program One7T90;
{ -- This program will print divisions in COBOL program. }
  var
    D: String[15];

begin
  Write ('Enter division: ');  Readln (D);
  if D = 'IDENTIFICATION' then begin
    Writeln ('BEFORE = NONE');
    Writeln ('AFTER = ENVIRONMENT  DATA  PROCEDURE');
    end
  else if D = 'ENVIRONMENT' then begin
    Writeln ('BEFORE = IDENTIFICATION');
    Writeln ('AFTER = DATA  PROCEDURE');
    end
  else if D = 'DATA' then begin
    Writeln ('BEFORE = IDENTIFICATION  ENVIRONMENT');
    Writeln ('AFTER = PROCEDURE');
    end
  else if D = 'PROCEDURE' then begin
    Writeln ('BEFORE = IDENTIFICATION  ENVIRONMENT  DATA');
    Writeln ('AFTER = NONE');
    end;
end.
```

```
{1.8}
program One8T90;
{ -- This program will display states having holidays. }
  var
    N: Byte;

begin
  Write ('Enter N: ');  Readln (N);
  if N <= 7 then
    Writeln ('FL NC SC TX MD GA VA')
  else if N = 8 then
    Writeln ('FL NC TX MD GA VA')
  else if (N = 9) or (N = 10) then
    Writeln ('FL TX MD GA VA')
  else if (N = 11) then
    Writeln ('MD');
end.


{1.9}
program One9T90;
{ -- This program will correct modern dates. }
  var
    Dat: Integer;
    AD:  String[4];

begin
  Write ('Enter date: ');  Readln (Dat);
  Write ('Enter A.D. or B.C.: ');  Readln (AD);
  if (AD = 'B.C.') and (Dat > 4) then
    Writeln (Dat - 4, ' B.C.')
  else if AD = 'B.C.' then
    Writeln (5 - Dat, ' A.D.')
  else
    Writeln (Dat + 4, ' A.D.');
end.


{1.10}
program One10T90;
{ -- This program will print a 7 letter word diamond. }
  var
    Word: String[7];

begin
  Write ('Enter word: ');  Readln (Word);
  Writeln (' ' :3, Copy(Word, 4, 1));
  Writeln (' ' :2, Copy(Word, 3, 3));
  Writeln (' ' :1, Copy(Word, 2, 5));
  Writeln (Word);
  Writeln (' ' :1, Copy(Word, 2, 5));
  Writeln (' ' :2, Copy(Word, 3, 3));
  Writeln (' ' :3, Copy(Word, 4, 1));
end.
```

```
{2.1}
program Two1T90;
{ -- This program will encode a phrase. }
  var
    A:  String[60];
    Ch: Char;
    I:  Byte;

begin
  Write ('Enter phrase: ');  Readln (A);
  for I := 1 to Length(A) do begin
    Ch := A[I];
    if (Ch < 'A') or (Ch > 'Z') then
      Write (Ch)
    else
      if Ch = 'A' then
        Write ('Z')
      else
        Write (Chr(Ord(Ch) - 1));
  end;
end.


{2.2}
program Two2T90;
{ -- This program will determine the "type" of year. }
  var
    Y: Integer;

begin
  Write ('Enter year: ');  Readln (Y);
  if Y mod 10   = 0 then Writeln ('END OF DECADE');
  if Y mod 100  = 0 then Writeln ('END OF CENTURY');
  if Y mod 1000 = 0 then Writeln ('END OF MILLENNIUM');
  if Y mod 10   = 1 then Writeln ('BEGINNING OF DECADE');
  if Y mod 100  = 1 then Writeln ('BEGINNING OF CENTURY');
  if Y mod 1000 = 1 then Writeln ('BEGINNING OF MILLENIUM');
end.
```

```
{2.3}
program Two3T90;
{ -- This program will print average and handicap of bowlers. }
  const
    A: Array [1..4] of String[8] =
       ('BOB:    ', 'DOUG:   ', 'JACKIE: ', 'JOSE:   ');
  var
    I, S1, S2, S3: Integer;
    Ave, Han:      Array [1..4] of Real;

begin
  for I := 1 to 4 do begin
    Write ('Enter scores for ', A[I]);  Readln (S1, S2, S3);
    Ave[I] := (S1 + S2 + S3) / 3.0;
    if Ave[I] > 200.0 then
      Han[I] := 0.0
    else
      Han[I] := (200.0 - Ave[I]) * 0.9;
  end;
  for I := 1 to 4 do begin
    Write (A[I], 'AVERAGE = ', Trunc(Ave[I] + 0.0001));
    Writeln ('  HANDICAP = ', Trunc(Han[I] + 0.0001));
  end;
end.


{2.4}
program Two4T90;
{ -- This program will determine # of days to add to date. }
  var
    Date:       String[10];
    MM, DD, YY: Integer;
    Code:       Integer;

begin
  Write ('Enter date: ');  Readln (Date);
  Val (Copy(Date,1,2), MM, Code);
  Val (Copy(Date,4,2), DD, Code);
  Val (Copy(Date,7,4), YY, Code);
  Write ('ADD ');
  if (YY < 1700) or ((YY = 1700) and (MM < 3)) then
    Writeln ('10 DAYS')
  else if (YY < 1800) or ((YY = 1800) and (MM < 3)) then
    Writeln ('11 DAYS')
  else if (YY < 1900) or ((YY = 1900) and (MM < 3)) then
    Writeln ('12 DAYS')
  else if (YY < 2100) or ((YY = 2100) and (MM < 3)) then
    Writeln ('13 DAYS');
end.
```

```
{2.5}
program Two5T90;
{ -- This program will sort efficiencies of sorting algorithms. }
  var
    N, I, J: Integer;
    Name:    Array [1..3] of String[11];
    A:       Array [1..3] of Real;
    X:       Real;
    T:       String[11];

begin
  Name[1] := 'BUBBLE SORT';  Name[2] := 'SHELL SORT';
  Name[3] := 'QUICK SORT';
  Write ('Enter N: ');  Readln (N);
  A[1] := N * (N-1) / 2;
  A[2] := N * (Ln(N) / Ln(2)) * (Ln(N) / Ln(2));
  A[3] := N * (Ln(N) / Ln(2));
  for I := 1 to 2 do
    for J := I+1 to 3 do
      if A[I] > A[J] then begin
        X := A[I];  A[I] := A[J];  A[J] := X;
        T := Name[I];  Name[I] := Name[J];  Name[J] := T;
      end;
  for I := 1 to 3 do Writeln (Name[I]);
end.
```

```
{2.6}
program Two6T90;
{ -- This program will determine status for each hole of golf. }
  const
    P: Array [1..9] of Byte = (4, 3, 4, 5, 4, 3, 5, 4, 4);
  var
    S: Array [1..9] of Byte;
    I, Sum, Par: Byte;
    D: Integer;

begin
  Sum := 0;  Par := 36;
  for I := 1 to 9 do begin
    Write ('Enter score for hole ', I, ': ');  Readln (S[I]);
    Sum := Sum + S[I];
  end;
  Writeln ('HOLE  PAR  SCORE  STATUS');
  Writeln ('----  ---  -----  ------');
  for I := 1 to 9 do begin
    Write (I:2, '     ', P[I], '     ', S[I], '    ');
    D := S[I] - P[I];
    case D of
     -3: Writeln ('DOUBLE EAGLE');
     -2: Writeln ('EAGLE');
     -1: Writeln ('BIRDIE');
      0: Writeln ('PAR');
      1: Writeln ('BOGEY');
      2: Writeln ('DOUBLE BOGEY');
    end;
  end;
  Writeln (' ':6, '---  -----');
  Writeln (' ':6, Par, '    ', Sum);
end.
```

```
{2.7}
program Two7T90;
{ -- This program will determine time calendar is ahead/behind. }
  var
    H, M, D, LY:                    Integer;
    N, Hour, Min, Sec, SN, MN, HN: Real;

begin
  Write ('Enter N: ');  Readln (N);
  { -- Sum 5 hours 48 min 47.8 sec for every year. }
  Hour := 5 * N;  Min := 48 * N;  Sec := 47.8 * N;
  { -- Convert to standard form }
  SN := Int(Sec  / 60);  Sec  := Sec  - SN*60;  Min := Min + SN;
  MN := Int(Min  / 60);  Min  := Min  - MN*60;  Hour:= Hour + MN;
  HN := Int(Hour / 24);  Hour := Hour - HN*24;  D   := Trunc(HN);
  H  := Trunc(Hour);     M    := Trunc(Min);
  { -- Subtract 1 for every leap year counted }
  LY := Trunc(N / 4);
  if LY <= D then begin
    Write (D - LY, ' DAYS  ', H, ' HOURS  ', M, ' MIN  ');
    Writeln (Sec:3:1, ' SEC  AHEAD'); end
  else begin
    Write ((LY - D - 1), ' DAYS  ', 23 - H, ' HOURS  ');
    Writeln (59 - M, ' MIN  ', 60 - Sec:3:1, ' SEC  BEHIND');
  end;
end.
```

```
{2.8}
program Two8T90;
{ -- This program will display members on a committee. }
  const
    A: Array [1..15] of String[10] = ('JACKIE', 'TOM',
       'LOVETTA', 'GREG', 'TONY', 'AL', 'KAREN', 'JAN', 'NORM',
       'TRUDY', 'THERESA', 'ALICE', 'DAVE', 'JIM', 'STEVE');
  var
    Y, Year:        Integer;
    I, M, J, Month: Byte;
    N:              Array [1..3] of String[10];
    NMonth:         Array [1..3] of Byte;

begin
  N[1] := 'BARB';  NMonth[1] := 6;
  N[2] := 'JOE';   NMonth[2] := 8;
  N[3] := 'DOUG';  NMonth[3] := 9;  Y := 1989;  M := 9;
  Write ('Enter month, year: ');  Readln (Month, Year);
  Writeln (M:2, '/', Y, ' - ', N[1], '  ', N[2], '  ', N[3]);
  I := 1;
  while (M <> Month) or (Y <> Year) do begin
    Inc(M);
    if M = 13 then begin
      M := 1;  Inc(Y);
    end;
    for J := 1 to 3 do
      if Abs(M - NMonth[J]) = 6 then begin
        N[J] := A[I];  Inc(I);  NMonth[J] := M;
        Write (M:2, '/', Y, ' - ');
        Writeln (N[1], '  ', N[2], '  ', N[3]);
      end;
  end;
end.
```

```
{2.9}
program Two9T90;
{ -- This program will graph the sine and cosine functions. }
uses Crt;
  var
    F, I, R, C:    Integer;
    X, CInc, RInc: Real;
    A:             String[1];

begin
  for F := 1 to 2 do begin
    ClrScr;
    for I := 1 to 24 do Writeln (' ': 39, '!');
    GotoXY (1, 12);
    for I := 1 to 79 do Write ('-');
    GotoXY (40, 12);  Write ('+');
    CInc := 39. / 3.14;  RInc := 11;
    For I := 0 to 628 do begin
      X := (I - 314) / 100;
      C := 40 + Round(CInc * X);
      if F = 1 then
        R := 12 - Round(Sin(X) * RInc)
      else
        R := 12 - Round(Cos(X) * RInc);
      GotoXY (C, R);  Write ('*');
    end;
    A := ''; while A = '' do A := ReadKey;
  end;
  ClrScr;
end.
```

```
{2.10}
program Two10T90;
{ -- This program will estimate hours of training given choices. }
uses Crt;
  const
    Low:  Array[1..7] of Real = (6.5, 4.5, 15, 4, 7, 6, 4);
    High: Array[1..7] of Byte = (8, 6, 20, 7, 11, 8, 6);
    A:    Array[1..7] of String[7] =
      ('187-11X', '187-15X', '220-AXX', '200-AXX', '123-2XX',
       '130-11X', '130-15X');
    B:    Array[1..7] of String[40] =
      ('ISPF/PDS FUNDAMENTALS     6.5 - 8',
       'ISPF/PDS FOR PROGRAMMERS  4.5 - 6',
       'JCL FUNDAMENTALS           15 - 20',
       'VSAM CONCEPTS               4 - 7',
       'MVS/SP/XA VSAM              7 - 11',
       'CICS/VS SKILLS I            6 - 8',
       'CICS/VS SKILLS II           4 - 6');
  var
    I, Num, HSum: Integer;
    CN:           Array [1..7] of Integer;
    LSum:         Real;
    C:            String[7];

begin
  ClrScr;
  Writeln ('           NCNB IN-HOUSE TRAINING LIST');
  Writeln;
  Writeln ('COURSE #   COURSE NAME               EST. HOURS');
  Writeln ('--------   -----------               ----------');
  for I := 1 to 7 do Writeln (A[I], '    ', B[I]);
  Writeln;  Num := 0;  LSum := 0;  HSum := 0;
  Write ('Enter course # (or 000-000 to end): ');  Readln (C);
  while C <> '000-000' do begin
    I := 1;  while C <> A[I] do Inc(I);
    Inc(Num);  CN[Num] := I;
    LSum := LSum + Low[I];  HSum := HSum + High[I];
    Write ('Enter course # (or 000-000 to end): ');  Readln (C);
  end;
  { -- Display options selected and TOTAL estimated hours. }
  ClrScr;
  Writeln ('COURSE NAME               EST. HOURS');
  Writeln ('-----------               ----------');
  for I := 1 to Num do Writeln (B[ CN[I] ]);
  Writeln ('                          ----------');
  Write   ('                TOTAL = ', LSum:4:1, ' - ');
  Writeln (HSum, ' HOURS');
end.
```

```
{3.1}
program Thr1T90;
{ -- This program will produce acronyms for phone numbers. }
  const
    A: Array [1..18] of String[5] = ('AGENT', 'SOAP', 'MONEY',
    'JEWEL', 'BALL', 'LOANS', 'CARE', 'SAVE', 'CALL', 'PAVE',
    'KEEP', 'KINGS', 'KNIFE', 'KNOCK', 'JOINT', 'JUICE',
    'LOBBY', 'RATE');
    L1: String[9] = ' ADGJMPTW';
    L2: String[9] = ' BEHKNRUX';
    L3: String[9] = ' CFILOSVY';
  var
    I, J, K, L: Integer;
    Ph, Num:    String[8];
    P4, P5:     String[5];
    C:          String[1];

begin
  Write ('Enter phone #: ');  Readln (Ph);
  P4 := Copy(Ph, 5, 4);  P5 := Copy(Ph, 3, 1) + P4;
  { -- Convert words to number strings }
  for I := 1 to 18 do begin
    L := Length(A[I]);  Num := '';
    for J := 1 to L do begin
      K := 2;  C := Copy(A[I], J, 1);
      while (L1[K] <> C) and (L2[K] <> C) and (L3[K] <> C) do
        Inc(K);
      Num := Num + Chr(48 + K);
    end;
    if (L = 4) and (Num = P4) then
      Writeln (Copy(Ph, 1, 4), A[I])
    else if (L=5) and (Num = P5) then begin
      Write (Copy(Ph, 1, 2),  Copy(A[I], 1, 1), '-');
      Writeln (Copy(A[I], L - 3, 4));
    end;
  end;
end.
```

```
{3.2}
program Thr2T90;
{ -- This program will select words given a string w/ wildcard. }
  const A: Array[1..25] of String[11] =
   ('COMPUTE', 'COMPUTER', 'COMPUTERS', 'COMPORT', 'COMPUTES',
    'COMPUTED', 'ATTRACTIVE', 'ABRASIVE', 'ADAPTIVE', 'ACCEPTIVE',
    'AERATING', 'CONTESTED', 'CONTESTER', 'CORONETS', 'CONTESTS',
    'CONTESTERS', 'COUNTESS', 'CREATIVE', 'CREATE', 'CREATURE',
    'CREATION', 'EVERYBODY', 'EVERYONE', 'EMPTY', 'ELECTION');
  var
    I, J, N, L, W: Byte;
    St, X, Ri, Le: String[11];

begin
  N := 25;
  repeat
    Write ('Enter string: ');  Readln (St);
    L := Length(St);  W := 0;  I := 0;  X := '';
    while (I <= L) and (X <> '*') do begin
      Inc(I);  X := Copy(St, I, 1);
    end;
    if I > L then Exit;
    { -- Asterisk is at position I }
    { -- Compare Left part of string and Right part of string. }
    Le := Copy(St, 1, I-1);  Ri := Copy (St, I+1, L-I);
    for J := 1 to N do
      if (Copy(A[J], 1, I-1) = Le) and
         (Copy(A[J], Length(A[J]) - (L-I) + 1, L-I) = Ri) then
        begin
          Write (A[J], '  ');  W := 1;
        end;

    if W = 0 then Writeln ('NO WORDS FOUND');
    Writeln;
  until I > L;
end.
```

```
{3.3}
program Thr3T90;
{ -- This program will keep score for a double dual race. }
uses Crt;
  var
    Init:                Array [1..21] of Char;
    TeamName:            Array [1..3] of Char;
    I, J, K:             Byte;
    StillUnique:         Boolean;
    UniqueTeams, Pl:     Byte;
    Team1Pos, Team2Pos: Array [1..7] of Byte;
    Team1,    Team2:     Byte;
    Team1Pl,  Team2Pl:   Byte;
```

```
begin
  ClrScr;  UniqueTeams := 0;
  for I := 1 to 21 do begin
    Write ('Place ', I: 2, ': ');  Readln (Init[I]);
    J := 0;  StillUnique := True;
    while (J < UniqueTeams) and StillUnique and (I > 1) do begin
      Inc(J);
      if TeamName[J] = Init[I] then
        StillUnique := False;
    end; { -- while }
    if StillUnique then
      begin
        Inc(UniqueTeams);
        TeamName[UniqueTeams] := Init[I];
      end;
  end; { -- for I }
  { -- Assert that Team[1,2,3] = 3 unique team Initials. }

  for I := 1 to 2 do
    for J := I+1 to 3 do begin
      PL := 0;  Team1 := 0;  Team2 := 0;
      Team1Pl := 0;  Team2Pl :=0;
      for K := 1 to 21 do begin
        if Init[K] = TeamName[I] then
        begin
          Inc(Pl);
          Team1 := Team1 + Pl;
          Inc(Team1Pl);
          Team1Pos[Team1Pl] := Pl
        end;
        if Init[K] = TeamName[J] then
        begin
          Inc(Pl);
          Team2 := Team2 + Pl;
          Inc(Team2Pl);
          Team2Pos[Team2Pl] := Pl
        end;
      end;  { -- for K }
      Team1 := Team1 - Team1Pos[6] - Team1Pos[7];
      Team2 := Team2 - Team2Pos[6] - Team2Pos[7];
      Writeln ('TEAM ', TeamName[I], ': ', Team1, ' POINTS');
      Writeln ('TEAM ', TeamName[J], ': ', Team2, ' POINTS');
      if (Team1 < Team2)
      or ((Team1 = Team2) and (Team1Pos[6] < Team2Pos[6])) then
        Write ('TEAM ', TeamName[I])
      else
        Write ('TEAM ', TeamName[J]);
      Writeln (' WINS!');  Writeln;
    end;  { -- for J }
end.
```

```
{3.4}
program Thr4T90;
{ -- This program will determine who gets which program #s. }
  const
    A: Array[1..8] of String[20] =
       ('AL, DOUG, AND JAN = ',
        'AL AND DOUG = ',
        'AL AND JAN = ',
        'DOUG AND JAN = ',
        'AL = ',
        'DOUG = ',
        'JAN = ',
        'NORM = ');
  var
    X, Y, Z, I, K:          Byte;
    XD, YD, ZD, One, Print: Boolean;

begin
 Write ('Enter X, Y, Z: ');  Readln (X, Y, Z);
 for K := 1 to 8 do begin
   Write (A[K]);
   One := False;
   for I := 1 to 30 do begin
    XD := (I/X = INT(I/X));  YD := (I/Y = INT(I/Y));
    ZD := (I/Z = INT(I/Z));  Print := False;
    if (K=1) and XD and YD and ZD then Print := True else
    if (K=2) and XD and YD and NOT ZD then Print := True else
    if (K=3) and XD and NOT YD and ZD then Print := True else
    if (K=4) and NOT XD and YD and ZD then Print := True else
    if (K=5) and XD and NOT YD and NOT ZD then Print := True else
    if (K=6) and NOT XD and YD and NOT ZD then Print := True else
    if (K=7) and NOT XD and NOT YD and ZD then Print := True else
    if (K=8) and NOT XD and NOT YD and NOT ZD then Print := True;
    if Print then begin
      Write (I, ' ');  One := True;
    end;
  end;
  if not One then Writeln ('NONE') else Writeln;
 end; { -- for K }
end.

{3.5}
program Thr5T90;
{ -- This program will display numbers 1-8 and a blank in a
  -- 3 x 3 array.  When a digit is pressed, it moves into the
  -- blank (if possible). }
uses Crt;
  var
    I, J, X, R1, R2, IndX, IndY: Byte;
    Digit, BlankX, BlankY:       Byte;
    A:                           Array [1..3, 1..3] of Byte;
    Valid:                       Boolean;
    DigSt:                       String[1];
    Code:                        Integer;
```

```
begin
  { -- Randomly place numbers in Array A. }
  Randomize;
  for I := 1 to 3 do
    for J := 1 to 3 do
      A[I,J] := (I-1)*3 + J-1;
  for I := 1 to 3 do
    for J := 1 to 3 do begin  { -- swap array values }
      R1 := Random(3) + 1;  R2 := Random(3) + 1;
      X := A[I,J];  A[I,J] := A[R1,R2];  A[R1,R2] := X;
    end;
  repeat
    { -- Display array }
    ClrScr;
    for I := 1 to 3 do begin
      for J := 1 to 3 do
        if A[I,J] > 0 then Write (A[I,J], '  ')
        else begin
          Write ('   ');
          BlankX := I;  BlankY := J;
        end;
      Writeln;
    end;

    { -- Accept valid digit or 9 }
    Valid := False;
    repeat
      DigSt := ''; while DigSt = '' do DigSt := ReadKey;
      Val(DigSt,Digit,Code);
      for I := 1 to 3 do
        for J := 1 to 3 do
          if Digit = A[I,J] then begin
            IndX := I;  IndY := J;
          end;
      if Abs(BlankX - IndX) + Abs(BlankY - IndY) = 1 then
        { -- adjacent }
        Valid := True;
    until Valid or (Digit = 9);

    if Valid then begin  { -- move digit in space }
      X := A[IndX,IndY];  A[IndX,IndY] := A[BlankX,BlankY];
      A[BlankX,BlankY] := X;
    end;
  until Digit = 9;  { -- 9 pressed }
end.


{3.6}
program Thr6T90;
{ -- This program will simulate the moves of a chess game. }
uses Crt;
  var
    A: Array [1..10] of String[50];
    I, L, WKR, WKC, BKR, BKC, R1, C1, R2, C2, Mov: Byte;
    M, Piec: String[5];
```

```
begin
  A[8] := 'BR1 BK1 BB1 BQ  BK  BB2 BK2 BR2   !  8';
  A[7] := 'BP1 BP2 BP3 BP4 BP5 BP6 BP7 BP8   !  7';
  A[6] := '                                  !  6';
  A[5] := '                                  !  5';
  A[4] := '                                  !  4';
  A[3] := '                                  !  3';
  A[2] := 'WP1 WP2 WP3 WP4 WP5 WP6 WP7 WP8   !  2';
  A[1] := 'WR1 WK1 WB1 WQ  WK  WB2 WK2 WR2   !  1';
  A[9] := '----------------------------------';
  A[10]:= ' A   B   C   D   E   F   G   H';
  ClrScr;  L := Length(A[1]);
  for I := 8 downto 1 do Writeln (A[I]);
  Writeln (A[9]);  Writeln (A[10]);
     { -- Location of 2 kings }
  WKR := 1;  WKC := 5;  BKR := 8;  BKC := 5;
  R2  := 0;  C2  := 0;  Mov := 0;

  while ((R2<>WKR) or (C2<>WKC)) and ((R2<>BKR) or (C2<>BKC)) do
  begin
    GotoXY (1, 12);  ClrEol;  GotoXY (1, 12);
    if Mov = 0 then begin
      Write ('Enter white move: ');  Readln (M); end
    else begin
      Write ('Enter black move: ');  Readln (M); end;
    { -- Convert moves to coordinates }
    C1 := Ord(M[1]) - 64;  R1 := Ord(M[2]) - 48;
    C2 := Ord(M[4]) - 64;  R2 := Ord(M[5]) - 48;
    { -- Move piece from 1 string to another and redisplay }
    Piec := Copy(A[R1], (C1-1)*4+1, 4);
    A[R2] := Copy(A[R2], 1, (C2-1)*4) + Piec +
             Copy(A[R2], C2*4+1, L-C2*4);
    GotoXY (1, 9-R2);  Writeln (A[R2]);
    { -- Remove piece from string by placing spaces and redisplay.}
    A[R1] := Copy(A[R1], 1, (C1-1)*4) + '    ' +
             Copy(A[R1], C1*4+1, L-C1*4);
    GotoXY (1, 9-R1);  Writeln (A[R1]);
    { -- If a king moved, store new location }
    if (R1 = WKR) and (C1 = WKC) then begin
      WKR := R2;  WKC := C2;  R2 := 0;  C2 := 0;
    end;
    if (R1 = BKR) and (C1 = BKC) then begin
      BKR := R2;  BKC := C2;  R2 := 0;  C2 := 0;
    end;
    if Mov = 0 then Mov := 1 else Mov := 0;
  end;  { -- while }
  GotoXY (1, 12);  Write ('CHECK MATE, ');
  if (R2 = WKR) and (C2 = WKC) then
    Writeln ('BLACK WON       ')
  else
    Writeln ('WHITE WON       ');
end.
```

```
{3.7}
program Thr7T90;
{ -- This program will print date of Easter and Lent in a year. }
  const
    M: Array[0..18] of Byte =
       (4, 4, 3, 4, 3, 4, 4, 3, 4, 4, 3, 4, 4, 3, 4, 3, 4, 4, 3);
    D: Array [0..18] of Byte =  (14, 3, 23, 11, 31, 18, 8, 28,
      16, 5, 25, 13, 2, 22, 10, 30, 17, 7, 27);
   MD: Array [1..3] of Byte = (31, 28, 31);
   Mo: Array [2..4] of String[8] = ('FEBRUARY', 'MARCH', 'APRIL');
  var
    I, Y, Key, Days, X, EDay, EMon, LDay, LMon: Integer;

begin
  Write ('Enter year: ');  Readln (Y);
  Key := Y mod 19;
  { -- Calculate # of days between 1,1,1970 and date }
  Days := (Y-1970) * 365 + (Y - 1968) div 4;
  for I := 1 to M[Key] - 1 do Days := Days + MD[I];
  Days := Days + D[Key];
  X := Days mod 7;
  { -- If X = 0-Wed, 1-Thu, 2-Fri, 3-Sat, 4-Sun, 5-Mon, 6-Tue }
  if X in [0..3] then EDay := D[Key] + (4-X)
    else EDay := D[Key] + (11-X);
  EMon := M[Key];
  if (M[Key] = 3) and (EDay > MD[3]) then begin
    EDay := EDay - MD[3];  EMon := EMon + 1;
  end;
  Writeln ('EASTER IS ON ', Mo[EMon], ' ', EDay);
  { -- Compute date of Lent }
  LMon := EMon - 1;
  LDay := MD[LMon] + EDay - 46;
  if LDay < 1 then begin
    LMon := LMon - 1;  LDay := LDay + MD[LMon];
  end;
  if (LMon = 2) and (Y mod 4 = 0) then Inc(LDay);
  Writeln ('LENT IS ON ', Mo[LMon], ' ', LDay);
end.
```

```
{3.8}
program Thr8T90;
{ -- This program will keep score for a bowler. }
uses Crt;
  var
    I, J, Fr, Len: Byte;
    A:             Array [1..10] of String[3];
    Md:            Char;
    Look, Sum:     Array [0..10] of Integer;
    AA:            Array [1..10,1..3] of Byte;

begin
  ClrScr;
  for I := 1 to 10 do begin
    Write ('Enter frame ', I, ': ');  Readln (A[I]);
  end;

  Writeln;
  Writeln ('-1- -2- -3- -4- -5- -6- -7- -8- -9- -10-');
  Writeln ('---!---!---!---!---!---!---!---!---!---!');
  for I := 1 to 10 do
    Write (A[I]: 3, '!');
  Writeln;

  { -- Assign values to A FRames according to X, /, or pins }
  for Fr := 1 to 10 do begin
    AA[Fr,2] := 0;
    for J := 1 to Length(A[Fr]) do begin
      Md := A[Fr,J];
      if Md = 'X' then
        begin
          AA[Fr,J] := 10;  Look[Fr] := 2;
        end
      else if Md = '/' then
        begin
          AA[Fr,J] := 10 - AA[Fr,J-1];  Look[Fr] := 1;
        end
      else
        if Md = '-' then
          AA[Fr,J] := 0
        else begin
          AA[Fr,J] := Ord(Md) - Ord('0');  Look[Fr] := 0;
        end;
    end;  { -- for J }
  end;  { -- for Fr }

  { -- Assign Frame values with Look ahead }
  Sum[0] := 0;
  for Fr := 1 to 10 do begin
    Sum[Fr] := Sum[Fr-1] + AA[Fr,1] + AA[Fr,2];
    if Look[Fr] > 0 then
      if Look[Fr] = 1 then  { -- A spare / needs 1 more value }
        if Fr = 10 then
          Sum[Fr] := Sum[Fr] + AA[Fr,3]
        else
```

```
          Sum[Fr] := Sum[Fr] + AA[Fr+1,1]
      else   { -- A strike X needs 2 more values }
        if Fr = 10 then
          Sum[Fr] := Sum[Fr] + AA[Fr,3]
        else
          begin
            Sum[Fr] := Sum[Fr] + AA[Fr+1,1] + AA[Fr+1,2];
            if Fr < 9 then
              if AA[Fr+1,1] = 10 then
                Sum[Fr] := Sum[Fr] + AA[Fr+2,1];
          end;

    Len := Trunc (Ln(Sum[Fr]) / Ln(10)) + 1;
    Write (Sum[Fr]: Len, '': 3-Len, '!');
  end;  { -- for Fr }
  Writeln;
  for I := 1 to 40 do Write ('-');
  Writeln;
end.
```

```
{3.9}
program Thr9T90;
{ -- This program will solve an N x N system of equations. }
  var
    C:              Array[1..5,1..6] of Real;
    N, Row, Col, R: Byte;
    Den, X:         Real;

begin
  { -- Enter values in C array }
  Write ('Enter N: ');  Readln (N);
  for Row := 1 to N do begin
    Writeln ('Enter coefficients for row', Row);
    for Col := 1 to N do begin
      Write ('Co', Col, ': ');
      Readln (C[Row,Col]);
    end;
    Write ('Enter constant: ');  Readln (C[Row, N+1]);
  end;

  { -- Make main diagonals all 1s with 0s to the left. }
  for Row := 1 to N do begin
    Den := C[Row, Row];
    for Col := Row to N+1 do
      C[Row, Col] := C[Row, Col] / Den;
    for R := Row+1 to N do begin
      X := C[R, Row];
      for Col := Row to N+1 do
        C[R,Col] := C[R,Col] - X * C[Row,Col];
    end;
  end;

{ -- Make 0s on the right of 1s on main diagonal, except consts. }
  for Row := N downto 1 do
    for R := Row-1 downto 1 do begin
      X := C[R, Row];
      for Col := Row to N+1 do
        C[R,Col] := C[R,Col] - X * C[Row,Col];
    end;

  { -- Display solution }
  Write ('(', C[Row,N+1] :1:0);
  for Row := 2 to N do begin
    Write (', ', C[Row,N+1] :1:0);
  end;
  Writeln (')');
end.
```

```
{3.10}
program Thr10T90;
{ -- This program will solve cryptorithms with two 2-letter addends
  -- and a 3-letter sum, using only the letters A, B, C, D, and E.}

  var
    St1, St2, St3:       String[3];
    Letters, Numbers:    String[7];
    FirstLet, UniqueLet: Array [1..7] of Byte;
    N1St, N2St, SumSt:   String[3];
    Ch:                  String[1];
    Solution, AtLeast1:  Boolean;
    I, J, N1, N2, Sum, NumLet: Byte;

begin
  Write ('Enter first addend: ');  Readln (St1);
  Write ('Enter second addend: '); Readln (St2);
  Write ('Enter sum: ');           Readln (St3);
  Letters := St1 + St2 + St3;  NumLet := 0;  AtLeast1 := False;

  { Put in FirstLet[] the index of the first occurence of letter.}
  for I := 1 to 7 do begin
    Ch := Copy(Letters, I, 1);
    FirstLet[I] := Pos(Ch, Letters);
    if FirstLet[I] = I then begin { -- This is a new letter. }
      Inc(NumLet);
      UniqueLet[NumLet] := I;
    end;
  end;

  for N1 := 10 to 98 do              { -- N1 must be 2 digits, >9 }
    for N2 := 100-N1 to 98 do begin { -- N2 must be 2 digits, >9 }
      Sum := N1 + N2;               { -- Sum must be 3 digits, >99}
      Str (N1, N1St);  Str (N2, N2St);  Str (Sum, SumSt);
      Numbers := N1St + N2St + SumSt;
      I := 1;  Solution := True;
      { -- Check if similar letters correspond to similar numbers.}
      repeat
        Ch := Copy(Numbers, I, 1);
        if Ch <> Copy (Numbers, FirstLet[I], 1) then
          Solution := False;
        Inc(I);
      until (I > 7) or not Solution;

      { -- Check if unique letters correspond to unique digits }
      for I := 1 to NumLet-1 do
        for J := I+1 to NumLet do
          if Numbers[UniqueLet[I]] = Numbers[UniqueLet[J]] then
            Solution := False;

      if Solution then begin  { -- Display solution }
        for I := 1 to NumLet do begin
          Write (Letters[UniqueLet[I]], ' = ');
          Writeln (Numbers[UniqueLet[I]]);
        end;
```

```
        Writeln;  AtLeast1 := True;  Exit;  { -- Only 1 needed }
      end;
    end;  { -  for N2 }
    if not AtLeast1 then
      Writeln ('NO SOLUTION POSSIBLE');
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '91 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T91;
{ -- This program will display a phrase as a rectangle. }
uses Crt;
  const
    A = 'COMPUTER CONTEST 1991';
  var
    I, L: Byte;

begin
  ClrScr;
  Writeln(A);
  L := Length(A);
  for I := 2 to L - 1 do begin
    GotoXY(1, I);  Write (Copy(A, I, 1));
    GotoXY(L, I);  Write (Copy(A, L-I+1, 1));
  end;
  Writeln;
  for I := L downto 1 do
    Write (Copy(A, I, 1));
end.


{1.2}
program One2T91;
{ -- This program will display 2 random #s and their sum. }
  var
    X, Y: Integer;

begin
  Randomize;
  X := Random(19) - 9;
  Y := Random(19) - 9;
  Writeln (X, ' + ', Y, ' = ', X + Y);
end.
```

```
{1.3}
program One3T91;
{ -- This program prints the total point score for a team. }
  var
    I, P, Sum: Byte;
    Nam:       String[20];

begin
  Sum := 0;
  Write ('Enter name: ');  Readln (Nam);
  for I := 1 to 3 do begin
    Write ('Enter # of ', I, ' point programs: ');
    Readln (P);
    Sum := Sum + P * I;
  end;
  Writeln (Nam, ' SCORED ', Sum, ' POINTS');
end.


{1.4}
program One4T91;
{ -- This program displays a spreadsheet. }
uses Crt;
  var
    I: Byte;

begin
  ClrScr;
  Writeln ('   A B C D E F G H I J K L M N O P Q R S T');
  for I := 1 to 20 do
    Writeln (I:2);
end.


{1.5}
program One5T91;
{ -- This program determines the number of teams competing. }
  var
    X: Integer;

begin
  Write ('Enter number of students: '); Readln (X);
  Writeln (X div 4, ' TEAMS');
end.
```

```
{1.6}
program One6T91;
{ -- This program displays a word twice intersecting at a letter.}
uses Crt;
  var
    A:    String[12];
    L:    String[1];
    X, I: Byte;

begin
  Write ('Enter word: ');  Readln (A);
  Write ('Enter letter: ');  Readln (L);
  X := Pos(L, A);
  ClrScr;
  GotoXY (1, X);  Writeln (A);
  for I := 1 to Length(A) do begin
    GotoXY (X, I);  Write (Copy(A, I, 1));
  end;
end.


{1.7}
program One7T91;
{ -- This program displays fields from an account key. }
  var
    A: String[20];

begin
  Write ('Enter account key: ');  Readln (A);
  Writeln ('ORGANIZATION ', Copy(A, 1, 3));
  Writeln ('BRANCH ',       Copy(A, 4, 3));
  Writeln ('DEALER ',       Copy(A, 7, 4));
  Writeln ('CLASS ',        Copy(A, 11, 3));
  Writeln ('UNIT ',         Copy(A, 14, 6));
end.


{1.8}
program One8T91;
{ -- This program displays the # of job steps in JCL. }
  var
    L: String[5];
    S: Byte;

begin
  Write ('Enter line: ');  Readln (L);  S := 0;
  while L <> '//' do begin
    if L = 'EXEC' then Inc(S);
    Write ('Enter line: ');  Readln (L);
  end;
  Writeln (S, ' JOB STEPS');
end.
```

```
{1.9}
program One9T91;
{ -- This program will replace MAN with PERSON. }
  var
    S: String[100];
    M: String[3];
    I: Byte;

begin
  Write ('Enter sentence: ');  Readln (S);
  for I := 1 to Length(S) do begin
    M := Copy(S, I, 3);
    if M = 'MAN' then begin
      Write ('PERSON');  I := I + 2;  end
    else if M = 'MEN' then begin
      Write ('PERSONS');  I := I + 2; end
    else
      Write (Copy(S, I, 1));
  end;
end.


{1.10}
program One10T91;
{ -- This program determines the winner of two computer teams. }
  var
    N1, N2: String[20];
    T1, T2, TI1, TI2: Integer;
    P1, P2, Pen1, Pen2, H1, H2, M1, M2: Byte;

begin
  Write ('Enter team name: ');  Readln (N1);
  Write ('Enter points, time, penalties: ');
  Readln (P1, T1, Pen1);
  Write ('Enter team name: ');  Readln (N2);
  Write ('Enter points, time, penalties: ');
  Readln (P2, T2, Pen2);

  if P1 > P2 then
    Write (N1)
  else if P2 > P1 then
    Write (N2)
  else begin
    H1 := T1 div 100;  M1 := T1 mod 100;
    H2 := T2 div 100;  M2 := T2 mod 100;
    TI1 := H1 * 60 + M1 + Pen1 * 5;
    TI2 := H2 * 60 + M2 + Pen2 * 5;
    if TI1 < TI2 then
      Write (N1)
    else
      Write (N2);
  end;
  Writeln (' WINS');
end.
```

```
{2.1}
program Two1T91;
{ -- This program displays a pyramid of consecutive numbers. }
  var
    N, S, I, J: Byte;

begin
  Write ('Enter N: ');  Readln (N);
  S := 1;  I := 0;
  while S < N do begin
    Inc(I);
    Write (' ': 20 - I * 2);
    for J := 1 to I do begin
      if S < 10 then Write ('0');
      Write (S, '  ');
      Inc(S);
    end;
    Writeln;
  end;
end.


{2.2}
program Two2T91;
{ -- This program will line up numbers with decimal points. }
  var
    I, X, Code: Integer;
    A:          Array [1..5] of String[9];
    Y, Sum:     Real;

begin
  for I := 1 to 5 do begin
    Write ('Enter #: ');  Readln (A[I]);
  end;
  Sum := 0;
  for I := 1 to 5 do begin
    X := Pos('.', A[I]);
    Writeln (' ': 6 - X, A[I]);
    Val(A[I], Y, Code);
    Sum := Sum + Y;
  end;
  Writeln (' ---------');
  Writeln (Sum: 10:4);
end.
```

```
{2.3}
program Two3T91;
{ -- This program will convert BASIC to COBOL. }
  var
    S:  String[80];
    M:  String[1];
    MN: String[2];
    I:  Byte;

begin
  Write ('Enter statement: ');  Readln (S);
  for I := 1 to Length(S) do begin
    M  := Copy(S, I, 1);
    MN := Copy(S, I, 2);
    if (MN = '<=') or (MN = '=<') then
      begin
        Write ('IS NOT GREATER THAN');
        Inc(I);
      end
    else if (MN = '>=') or (MN = '=>') then
      begin
        Write ('IS NOT LESS THAN');
        Inc(I);
      end
    else if (MN = '<>') or (MN = '><') then
      begin
        Write ('IS NOT EQUAL TO');
        Inc(I);
      end
    else if (M = '>') then
      Write ('IS GREATER THAN')
    else if (M = '<') then
      Write ('IS LESS THAN')
    else if (M = '=') then
      Write ('IS EQUAL TO')
    else
      Write (M);
  end;
end.
```

```
{2.4}
program Two4T91;
{ -- This program ranks teams in a league. }
  var
    N, I, J, R, X: Integer;
    Na:   Array [1..9] of String[20];
    W, L: Array [1..9] of Integer;
    T:    String[20];

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter team: ');  Readln(Na[I]);
    Write ('Enter wins, losses: ');  Readln (W[I], L[I]);
  end;
  for I := 1 to N - 1 do
    for J := I + 1 to N do
      if (W[I] < W[J]) or ((W[I] = W[J]) and (Na[I] > Na[J])) then
      begin
        X := W[I];  W[I] := W[J];  W[J] := X;
        X := L[I];  L[I] := L[J];  L[J] := X;
        T := Na[I]; Na[I]:=Na[J]; Na[J] := T;
      end;
  for I := 1 to N do begin
    if W[I] = W[I - 1] then
      Write (R)
    else begin
      Writeln;  Write(I);  R := I;
    end;
    Write (' ', Na[I], ' ': 14 - Length(Na[I]), W[I]);
    Writeln (' , ', L[I]);
  end;
end.

{2.5}
program Two5T91;
{ -- This program will guess a secret number within 7 tries. }
  var
    Increment, Guess, G: Byte;
    A: Char;

begin
  Increment := 64;  Guess := 64;  G := 0; A := ' ';
  while A <> 'R' do begin
    Inc(G);
    Writeln ('GUESS ', G, ': ', Guess);
    Write ('Enter H, L, or R: ');  Readln (A);
    Increment := Increment div 2;
    if A = 'L' then
      Dec(Guess, Increment);
    if A = 'H' then
      Inc(Guess, Increment);
  end;
end.
```

```
{2.6}
program Two6T91;
{ -- This program prints text in pyramid form. }
  var
    A, Lin:   String[255];
    I, L, PL: Byte;
    MD:       String[1];

begin
  Write ('Enter text: ');  Readln (A);
  L := Length(A);  I := 1;  PL := 0; Lin := '';
  while I <= L do begin
    MD := Copy(A, I, 1);
    if MD <> ' ' then
      Lin := Lin + MD
    else if Length(Lin) < PL + 2 then
      Lin := Lin + MD
    else begin
      PL := Length(Lin);
      Writeln (' ': 20 - (PL div 2), Lin);
      Lin := '';
    end;
    Inc(I);
  end;
  PL := Length(Lin);
  Writeln (' ': 20 - (PL div 2), Lin);
end.


{2.7}
program Two7T91;
{ -- This program displays a rectangle of asterisks. }
uses Crt;
  var
    L, W, I, Row, Col: Byte;

begin
  Write ('Enter length, width: '); Readln (L, W);
  ClrScr;
  Col := (80 - L) div 2;  Row := (24 - W) div 2;
  GotoXY (Col, Row);
  for I := 1 to L do Write ('*');
  for I := 1 to W - 2 do begin
    GotoXY (Col, Row + I);  Write ('*');
    GotoXY (Col + L - 1, Row + I);  Write ('*');
  end;
  GotoXY (Col, Row + W - 1);
  for I := 1 to L do Write ('*');
end.
```

```
{2.8}
program Two8T91;
{ -- This program displays a bar graph for lengths. }
uses Crt;
  var
    A:    Array [0..11] of Integer;
    I, J: Byte;
    Max:  Integer;
    Inc:  Real;
    T:    String[40];

begin
  Write ('Enter title: ');  Readln (T);  Max := 0;
  for I := 0 to 11 do begin
    Write ('Enter # for ', 1980 + I, ': ');  Readln (A[I]);
    if A[I] > Max then Max := A[I];
  end;
  Inc := Max / 20.0;
  ClrScr;
  Writeln (' ': 3, T, ' ': 3, 'ASTERISK = ', Inc: 7:2);
  for I := 20 downto 1 do Writeln (I: 2);
  for I := 1 to 12 * 3 + 2 do Write ('-');
  Writeln;  Write (' ': 2);
  for I := 0 to 11 do Write (80 + I: 3);
  for I := 0 to 11 do
    for J := 1 to Trunc(A[I] / Inc) do begin
      GotoXY (I * 3 + 5, 22 - J);  Write ('*');
    end;
  GotoXY(1, 22);
end.



{2.9}
program Two9T91;
{ -- This program displays a store maintenance list. }
  var
    I, I1, I2, F1, F2: Byte;
    AN, CN, DN:        Byte;
    A, C, D:           Array [1..9] of String[10];
    ID1, ID2:          Array [1..9] of String[4];
    Item1, Item2:      Array [1..9] of Char;

begin
  Write ('Enter # of entries in yesterday''s file: ');
  Readln (F1);
  for I := 1 to F1 do begin
    Write ('Enter ID: ');  Readln (ID1[I]);
    Write ('Enter item: ');  Readln (Item1[I]);
  end;
  Write ('Enter # of entries in today''s file: ');
  Readln (F2);
  for I := 1 to F2 do begin
    Write ('Enter ID: ');  Readln (ID2[I]);
    Write ('Enter item: ');  Readln (Item2[I]);
```

```
  end;
  ID2[F2 + 1] := 'ZZZZ';  ID1[F1 + 1] := '    ';
  I1 := 1;  I2 := 1;  AN := 0;  CN := 0;  DN := 0;
  while (I1 <= F1) or (I2 <= F2) do
    if ID1[I1] = ID2[I2] then
      if Item1[I1] <> Item2[I2] then  { -- Changed }
        begin
          Inc(CN);
          C[CN] := ID1[I1] + ' ' + Item1[I1] + ' ' + Item2[I2];
          Inc(I1);  Inc(I2);
        end
      else  { -- No change }
        begin
          Inc(I1);  Inc(I2);
        end
    else
      if (ID1[I1] < ID2[I2]) and (I1 <= F1) then  { -- Deleted }
        begin
          Inc(DN);
          D[DN] := ID1[I1] + ' ' + Item1[I1];
          Inc(I1);
        end
      else
        begin  { -- Added }
          Inc(AN);
          A[AN] := ID2[I2] + ' ' + Item2[I2];
          Inc(I2);
        end;

  Writeln;  Writeln ('ADDED');
  for I := 1 to AN do Writeln (A[I]);
  Writeln;  Writeln ('CHANGED');
  for I := 1 to CN do Writeln (C[I]);
  Writeln;  Writeln ('DELETED');
  for I := 1 to DN do Writeln (D[I]);
  Writeln;
  Writeln ('TOTAL ADDED = ', AN);
  Writeln ('TOTAL CHANGED = ', CN);
  Writeln ('TOTAL DELETED = ', DN);
end.
```

```
{2.10}
program Two10T91;
{ -- This program displays the contents of contest diskettes. }
uses Crt;
  const
    Z: Array [1..6] of String[3] =
       ('PRB', 'JDG', 'PG1', 'PG2', 'BAS', 'PAS');
    X: Array [1..3] of String[3] = ('ONE', 'TWO', 'THR');
  var
    I, J, K, P, Y, Tot: Byte;
    Year: String[4];
    YY:   String[2];
    Ch:   Char;

begin
  Write ('Enter year: ');  Readln (Year);
  YY := Copy(Year, 3, 2);
  for I := 1 to 4 do
    for J := 1 to 3 do
      Writeln ('FHS', YY, '-', J, '.', Z[I]);

  Tot := 12;
  for I := 5 to 6 do
    for J := 1 to 3 do begin
      P := 10;
      if (YY = '80') and (J = 3) then P := 12;
      if (YY = '81') then P := 5;
      if (YY = '82') and (J = 2) then P := 12;
      if (YY = '82') and (J = 3) then P := 8;
      for K := 1 to P do begin
        Writeln (X[J], K, 'T', YY, '.', Z[I]);
        Inc(Tot);
        if Tot = 20 then begin
          Ch := ReadKey;
          Tot := 0;
        end;
      end;
    end;  { -- for J }

end.
```

```
{3.1}
program Thr1T91;
{ -- This program simulates a baseball game. }
uses Crt;
  var
    I, Inn, T, S, B, W, R, O, Wtot, Otot: Byte;
    Stot, Btot: Integer;
    Run:        Array [1..2] of Byte;

begin
  Randomize;  ClrScr;  Writeln;  Write (' ': 7);
  for I := 1 to 9 do Write (I:3);
  Writeln ('  SCORE');
  Write (' ': 8);
  for I := 1 to 34 do Write ('-');
  Writeln;
  Writeln ('TEAM A !', ' ': 27, '!');
  Writeln ('TEAM B !', ' ': 27, '!');
  Stot := 0;  Btot := 0;  Otot := 0;  Wtot := 0;
  Run[1] := 0;  Run[2] := 0;

  for Inn := 1 to 9 do
    for T := 1 to 2 do begin
      S := 0;  B := 0;  W := 0;  R := 0;  O := 0;
      while O < 3 do begin
        if Random < 0.4 then begin
          Inc(S);  Inc(Stot);  end
        else begin
          Inc(B);  Inc(Btot);
        end;
        if S = 3 then begin
          Inc(O);  Inc(Otot);  S := 0;  W := 0;
        end;
        if B = 4 then begin
          Inc(W);  Inc(Wtot);  B := 0;  S := 0
        end;
        if W = 4 then begin
          Inc(R);  Inc(Run[T]);  W := 3;
        end;
      end;
      GotoXY (6 + Inn * 3, 3 + T);  Write (R:2);
    end;  { -- for T }

  GotoXY (38, 4);  Writeln (Run[1]: 3);
  GotoXY (38, 5);  Writeln (Run[2]: 3);
  Writeln;
  Writeln ('TOTAL # OF STRIKES: ', Stot);
  Writeln ('TOTAL # OF BALLS: ', Btot);
  Writeln ('TOTAL # OF WALKS: ', Wtot);
  Writeln ('TOTAL # OF STRIKE OUTS: ', Otot);
end.
```

```
{3.2}
program Thr2T91;
{ -- This program displays the units digit in a power expression.}
  var
    A, X: Array [1..3] of Integer;
    I, J, Pow, Sum, C:    Integer;

begin
  Write ('Enter A, X: ');  Readln (A[1], X[1]);
  Write ('Enter B, Y: ');  Readln (A[2], X[2]);
  Write ('Enter C, Z: ');  Readln (A[3], X[3]);
  Sum := 0;
  for I := 1 to 3 do begin
    Pow := 1;
    for J := 1 to X[I] do begin
      Pow := Pow * A[I];
      C := Pow div 10;
      Pow := Pow - C * 10;
    end;
    Sum := Sum + Pow;
  end;
  C := Sum div 10;
  Writeln (Sum - C * 10);
end.


{3.3}
program Thr3T91;
{ -- This program displays all digits in X ^ Y. }
  var
    A: Array [1..200] of Integer;
    X, Y, I, J, Dig, C, CC: Integer;

begin
  Write ('Enter X, Y: ');  Readln (X, Y);
  Dig := 1;  A[1] := 1;  C := 0;
  for I := 1 to Y do begin
    for J := 1 to Dig do begin
      A[J] := A[J] * X + C;
      C := A[J] div 10;
      A[J] := A[J] - C * 10;
    end;
    while C > 0 do begin
      CC := C div 10;
      Dig := Dig + 1;
      A[Dig] := C - CC * 10;
      C := CC;
    end;
  end;
  for I := Dig downto 1 do Write (A[I]);
end.
```

```
{3.4}
program Thr4T91;
{ -- This program assigns user LOGON IDs to names. }
  var
    N, Fn, Mn, Ln, Init, In2, N2: Array [1..9] of String[20];
    T, I, J, M, F, Y, A, B:       Byte;
    C:                            Array [1..9] of Byte;
    MD:                           String[1];
    W, X:                         String[20];

begin
  Write ('Enter name: ');  Readln (N[1]);  T := 1;
  while N[T] <> 'END' do begin
    Inc(T);
    Write ('Enter name: ');  Readln (N[T]);
  end;
  { -- Extract parts of name for initials }
  Dec(T);
  for I := 1 to T do begin
    W := '';  M := 0;  F := 0;
    for J := 1 to Length(N[I]) do begin
      MD := Copy (N[I], J, 1);
      if MD <> ' ' then
        W := W + MD
      else
        if F = 1 then begin
          Mn[I] := W;  M := 1; W := '';  end
        else begin
          Fn[I] := W;  F := 1; W := ''; end;
    end;  { -- for J }
    if M = 0 then Mn[I] := 'X';
    Ln[I] := W;
    Init[I] := Copy(Fn[I],1,1) + Copy(Mn[I],1,1) + Copy(Ln[I],1,1);
    In2[I] := Init[I];  N2[I] := Ln[I] + ' ' + Fn[I];  C[I] := I;
  end;  { -- for I }
  { -- Sort Initials }
  for I := 1 to T - 1 do
    for J := I + 1 to T do
      if In2[I] > In2[J] then begin
        X := In2[I];  In2[I] := In2[J];  In2[J] := X;
        X := N2[I];   N2[I]  := N2[J];   N2[J]  := X;
        Y := C[I];    C[I]   := C[J];    C[J]   := Y;
      end;
  { -- Sort names within same initials and assign numbers. }
  J := 0;
  while J < T - 1 do begin
    I := J + 1;  J := I + 1;
    while (In2[I] <> In2[J]) and (I < T) do begin
      Inc(I);  Inc(J);
    end;
    while (In2[I] = In2[J]) do  Inc(J);
    Dec(J);
    for A := I to J - 1 do
      for B := A + 1 to J do
        if N2[A] > N2[B] then begin
```

```
          X := N2[A];  N2[A] := N2[B];  N2[B] := X;
          Y := C[A];   C[A]  := C[B];   C[B]  := Y;
        end;
    { -- Assign numbers for middle initial }
    for A := I to J do
      Init[C[A]] := Copy(Init[C[A]],1,1) + Chr(48 + (A - I + 1))
                  + Copy(Init[C[A]],3,1);
  end;  { -- while }
  for I := 1 to T do
    Writeln (N[I], ' ': 19 - Length(N[I]), 'SD', Init[I], '1');
end.
```

```
{3.5}
program Thr5T91;
{ -- This program displays the digits 0 - 9 in enlarged form. }
{     1    The data contains the          }
{    2 3   line segment #s (on the left)  }
{     4    that need to be displayed to   }
{    5 6   produce the corresponding      }
{     7    digits: 0,1,2,3,4,5,6,7,8,9.   }
uses Crt;
  const
    A: Array [0..9] of String[7] =
        ('123567', '36', '13457', '13467', '2346',
         '12467', '124567', '136', '1234567', '12346');
  var
    N, I, J, X: Byte;

begin
  for N := 0 to 9 do begin
    ClrScr;
    for J := 1 to Length(A[N]) do begin
      X := Ord(A[N,J]) - Ord('0');
      Case X of
       1: begin
            GotoXY (1,1); for I := 1 to 11 do Write ('*');
          end;
       2: for I := 1 to 8 do begin
            GotoXY (1, I);  Write ('*');
          end;
       3: for I := 1 to 8 do begin
            GotoXY (11, I);  Write ('*');
          end;
       4: begin
            GotoXY (1,8); for I := 1 to 11 do Write ('*');
          end;
       5: for I := 1 to 8 do begin
            GotoXY (1, I+7);  Write ('*');
          end;
       6: for I := 1 to 8 do begin
            GotoXY (11, I+7);  Write ('*');
          end;
       7: begin
            GotoXY (1, 15);  for I := 1 to 11 do Write ('*');
          end;
      end;  { -- case }
     end;  { -- next J }
     Delay (1000);
   end;  { -- next N }
 end.
```

```
{3.6}
program Thr6T91;
{ -- This program will evaluate an expression with (). }
  var
    I, J, N, S, P: Byte;
    A:       String[50];
    Ch:      Char;
    P1, Num: Array [1..10] of Integer;
    SY:      Array [1..9]  of String[1];

begin
  Write ('Enter expression: ');  Readln (A);
  P := 0;  S := 0;  N := 0;
  for I := 1 to Length(A) do begin
    Ch := A[I];
    if Ch = '(' then begin
      Inc(P);  P1[P] := S + 1;  end
    else if (Ch = '+') or (Ch = '-') then begin
      Inc(S);  SY[S] := Ch; end
    else if (Ch >= '0') and (Ch <= '9') then begin
      Inc(N);  Num[N] := Ord(Ch) - 48;  end
    else if Ch = ')' then begin
      for J := P1[P] to S do begin
        if SY[J] = '-' then Num[J+1] := Num[J] - Num[J+1];
        if SY[J] = '+' then Num[J+1] := Num[J] + Num[J+1];
      end;
      N := P1[P];  Num[N] := Num[S + 1];
      S := P1[P] - 1;  Dec(P);
    end;
  end;
  for I := 1 to S do begin
    if SY[I] = '-' then Num[I+1] := Num[I] - Num[I+1];
    if SY[I] = '+' then Num[I+1] := Num[I] + Num[I+1];
  end;
  Writeln (Num[N]);
end.
```

```
{3.7}
program Thr7T91;
{ -- This program displays the two pay days for a given month. }
  const
    Mname: Array [1..12] of String[9] = ('JANUARY', 'FEBRUARY',
       'MARCH', 'APRIL', 'MAY', 'JUNE', 'JULY,', 'AUGUST',
       'SEPTEMBER', 'OCTOBER', 'NOVEMBER', 'DECEMBER');
    Mon: Array [1..12] of Byte =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    Dname: Array [1..7] of String[9] = ('MONDAY', 'TUESDAY',
       'WEDNESDAY', 'THURSDAY', 'FRIDAY', 'SATURDAY', 'SUNDAY');
  var
    I, T, H, Hol, Wkend, X, MNum: Byte;
    Mhol, Dhol: Array [1..12] of Byte;
    Day, Days:  Array [1..2]  of Integer;

begin
  H := 1;
  Write ('Enter holiday MM, DD: ');  Readln (Mhol[H], Dhol[H]);
  while Mhol[H] > 0 do begin
    Inc(H);
    Write ('Enter holiday MM, DD: ');  Readln (Mhol[H], Dhol[H]);
  end;
  Dec(H);  Writeln;
  Write ('Enter month #: ');  Readln (MNum);  Writeln;
  while MNum > 0 do begin
    Days[1] := 0;
    for I := 1 to MNum - 1 do
      Days[1] := Days[1] + Mon[I];
    Day[1] := 15;  Day[2] := Mon[MNum];
    Days[2] := Days[1] + Day[2];
    Days[1] := Days[1] + Day[1];
    for T := 1 to 2 do begin
      Hol := 1;  Wkend := 1;
      { -- Decrement days counter if holiday or weekend. }
      while (Hol = 1) or (Wkend = 1) do begin
        Hol := 0;  Wkend := 0;
        for I := 1 to H do
          if (Mhol[I] = MNum) and (Dhol[I] = Day[T]) then begin
            Dec(Day[T]);
            Dec(Days[T]);  Hol := 1;
          end;
        X := Days[T] mod 7;
        if (X = 5) or (X = 6) then begin  { -- Sat. or Sun. }
          Dec(Day[T]);
          Dec(Days[T]);  Wkend := 1;
        end;
      end;  { -- while }
      Writeln (Dname[X+1], ' ', Mname[MNum], ' ', Day[T]);
    end;  { -- for T }
    Writeln;  Write ('Enter month #: ');  Readln (Mnum);  Writeln;
  end;  { -- while }
end.
```

```
{3.8}
program Thr8T91;
{ -- This program will display 3 x 3 magic squares. }
  var
    Dig, Row, Col, I, J, P, Rot, X: Byte;
    A: Array [1..3,1..3] of Byte;

begin
  A[1,1] := 6;  A[1,2] := 7;  A[1,3] := 2;
  A[2,1] := 1;  A[2,2] := 5;  A[2,3] := 9;
  A[3,1] := 8;  A[3,2] := 3;  A[3,3] := 4;
  Write ('Enter digit: ');  Readln (Dig);
  Write ('Enter row, col: ');  Readln (Row, Col);
  Rot := 1;
  while (A[Row,Col] <> Dig) and (Rot < 4) do begin
    { -- Rotate outer numbers clockwise, at most 3 times }
    X := A[1,1];  A[1,1] := A[3,1];  A[3,1] := A[3,3];
    A[3,3] := A[1,3];  A[1,3] := X;
    X := A[1,2];  A[1,2] := A[2,1];  A[2,1] := A[3,2];
    A[3,2] := A[2,3];  A[2,3] := X;
    Inc(Rot);
  end;
  if A[Row,Col] <> Dig then begin
    Writeln ('NO SOLUTION');  Exit;
  end;
  for P := 1 to 2 do begin
    for I := 1 to 3 do begin
      for J := 1 to 3 do
        Write (A[I,J], '  ');
      Writeln;
    end;
    Writeln;
    if P = 1 then begin
       if (Row = 1) and (Col = 3) or (Row = 3) and (Col = 1) then
         begin
           X := A[2,1];  A[2,1] := A[3,2];  A[3,2] := X;
           X := A[1,1];  A[1,1] := A[3,3];  A[3,3] := X;
           X := A[1,2];  A[1,2] := A[2,3];  A[2,3] := X;
         end;
       if (Row = 1) and (Col = 1) or (Row = 3) and (Col = 3) then
         begin
           X := A[1,2];  A[1,2] := A[2,1];  A[2,1] := X;
           X := A[1,3];  A[1,3] := A[3,1];  A[3,1] := X;
           X := A[3,2];  A[3,2] := A[2,3];  A[2,3] := X;
         end;
       if (Row = 1) and (Col = 2) or (Row = 3) and (Col = 2) then
         begin
           X := A[1,1];  A[1,1] := A[1,3];  A[1,3] := X;
           X := A[2,1];  A[2,1] := A[2,3];  A[2,3] := X;
           X := A[3,1];  A[3,1] := A[3,3];  A[3,3] := X;
         end;
       if (Row = 2) and (Col = 1) or (Row = 2) and (Col = 3) then
         begin
           X := A[1,1];  A[1,1] := A[3,1];  A[3,1] := X;
           X := A[1,2];  A[1,2] := A[3,2];  A[3,2] := X;
```

```
          X := A[1,3];  A[1,3] := A[3,3];  A[3,3] := X;
        end;
    end;
  end;  { -- for P }
end.
```

```
{3.9}
program Thr9T91;
{ -- This program will display a pie graph. }
uses Crt;
  const
    L:  Array [1..3] of Char = ('A', 'D', 'N');
    PI: Real = 3.1415926;
  var
    A:  Array[1..21, 1..21] of Byte;
    P:  Array[1..3] of Byte;
    I:  Real;
    Ch: Char;
    J, K, R, X, Y, S, Sum, LSum: Integer;

begin
  Write ('Enter 3 percentages: ');  Readln (P[1], P[2], P[3]);
  ClrScr;
  for J := 1 to 21 do
    for K := 1 to 21 do
      A[J, K] := 0;
  { -- Draw Circle }
  I := -PI / 2.0;
  while I < 3 / 2 * PI do begin
    X := Trunc(Cos(I) * 10);  Y := Trunc(Sin(I) * 10);
    GotoXY (11 + X, 11 + Y);  Write ('*');
    A[11 + X, 11 + Y] := 1;  I := I + 0.1;
  end;
  { -- Draw 3 line segments from center }
  Sum := 0;
  for S := 0 to 2 do begin
    Sum := Sum + P[S];
    I := -PI / 2 + 2 * PI * Sum / 100.0;
    for R := 0 to 10 do begin
      X := Trunc(Cos(I) * R);  Y := Trunc(Sin(I) * R);
      GotoXY (11 + X, 11 + Y);  Write ('*');
      A[11 + X, 11 + Y] := 1;
    end;
  end;
  Ch := ReadKey;  Sum := 0;
  { -- fill regions with letters }
  for S := 1 to 3 do begin
    LSum := Sum;  Sum := Sum + P[S];  J := LSum;
    while J < Sum do begin
      I := -PI / 2 + 2 * PI * J / 100.0;
      for R := 1 to 9 do begin
        X := Trunc(Cos(I) * R);  Y := Trunc(Sin(I) * R);
        if A[11 + X, 11 + Y] = 0 then begin
          GotoXY (11 + X, 11 + Y);  Write (L[S]);
        end;
      end;
      Inc(J);
    end;
  end;
end.
```

```
{3.10}
program Thr10T91;
{ -- This program will convert large numbers in base 2,4,8,16. }
  var
    A:     Array [1..255] of Byte;
    D:     String[1];
    NumSt: String[65];
    I, J, K, L, M, N, X, Num, DigN,
    DigM, Pad, Ind, Pow, LInd, Zero, Sum: Byte;

begin
  Write ('Enter numeral: ');  Readln (NumSt);
  Write ('Enter base M: ');   Readln (M);
  Write ('Enter base N: ');   Readln (N);
  L := Length (NumSt);
  DigM := Trunc (Ln (M) / Ln (2) + 0.001);
  DigN := Trunc (Ln (N) / Ln (2) + 0.001);
  Pad := DigN - (DigM * L mod DigN);
  if Pad = DigN then Pad := 0;
  for I := 1 to Pad do  A[I] := 0;
  { -- Convert from base M to base 2 }
  for I := 1 to L do begin
    D := Copy (NumSt, I, 1);
    Num := Pos (D, '0123456789ABCDEF') - 1;
    for J := DigM - 1 downto 0 do begin
      Pow := 1;
      for K := 1 to J do Pow := Pow * 2;
      X := Num div Pow;
      Ind := I * DigM - J + Pad;
      A[Ind] := X;
      Num := Num - X * Pow;
    end;
  end;
  { -- Convert from base 2 to base N }
  LInd := DigM * L + Pad;  Zero := 1;
  for I := 0 to (Lind div Dign) - 1 do begin
    Sum := 0;
    for J := 1 to DigN do begin
      Ind := I * DigN + J;
      Pow := 1;
      for K := 1 to (DigN - J) do  Pow := Pow * 2;
      Sum := Sum + A[Ind] * Pow;
    end;
    if (Zero = 0) or (Sum > 0) then begin
      Zero := 0;
      Write (Copy ('0123456789ABCDEF', Sum + 1, 1));
    end;
  end;
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '92 }
{ -- PACSCAL PROGRAM SOLUTIONS }


{1.1}
program One1T92;
{ -- This program displays the company name: GTEDS. }

begin
  Writeln ('GGGGG   TTTTT   EEEEE');
  Writeln ('G         T     E');
  Writeln ('G GGG     T     EEEEE   DATA SERVICES');
  Writeln ('G   G     T     E');
  Writeln ('GGGGG     T     EEEEE');
end.


{1.2}
program One2T92;
{ -- This program will display the company name in a year. }
  var
    Year: Integer;

begin
  Write ('Enter year: ');  Readln (Year);
  if Year < 1920 then
    Writeln ('RICHLAND CENTER TELEPHONE COMPANY')
  else if Year < 1926 then
    Writeln ('COMMONWEALTH TELEPHONE COMPANY')
  else if Year < 1935 then
    Writeln ('ASSOCIATED TELEPHONE UTILITIES COMPANY')
  else if Year < 1959 then
    Writeln ('GENERAL TELEPHONE CORPORATION')
  else if Year < 1982 then
    Writeln ('GENERAL TELEPHONE & ELECTRONICS CORPORATION')
  else
    Writeln ('GTE CORPORATION');
end.


{1.3}
program One3T92;
{ -- This program will determine company's ranking in Forbes. }
  var
    Rank, Places: Integer;

begin
  Write ('Enter 1991 rank: ');  Readln (Rank);
  Write ('Enter number of places: ');  Readln (Places);
  Writeln (Rank - Places);
end.
```

```
{1.4}
program One4T92;
{ -- This program will indent GTE's 6 operations. }
  var
    X: Byte;
begin
  Write ('Enter number of spaces: ');  Readln (X);
  Writeln ('GTE TELEPHONE OPERATIONS');
  Writeln (' ': X, 'GTE GOVERNMENT SYSTEMS');
  Writeln (' ': X * 2, 'GTE MOBILE COMMUNICATIONS');
  Writeln (' ': X * 3, 'GTE INFORMATION SERVICES');
  Writeln (' ': X * 4, 'GTE SPACENET');
  Writeln (' ': X * 5, 'GTE AIRFONE');
end.


{1.5}
program One5T92;
{ -- This program will display # of WHOLE YEARS GTEDS existed. }
  var
    Month, Year, X: Integer;

begin
  Write ('Enter month, year: ');  Readln (Month, Year);
  if Month < 10 then
    X := 1
  else
    X := 0;
  Writeln (Year - 1967 - X, ' YEARS');
end.


{1.6}
program One6T92;
{ -- This program will center a title and name in a box. }
  var
    Title, Name:    String[20];
    I, L, Sp1, Sp2: Byte;

begin
  Write ('Enter title: ');  Readln (Title);
  Write ('Enter name: ');  Readln (Name);
  for I := 1 to 24 do Write ('*');
  Writeln;
  Writeln ('*', ' ': 22, '*');

  L := Length(Title) + Length(Name) + 1;
  Sp1 := (22 - L) div 2;
  Sp2 := (22 - L) - Sp1;
  Writeln ('*', ' ': Sp1, Title, ' ', Name, ' ': Sp2, '*');

  Writeln ('*', ' ': 22, '*');
  for I := 1 to 24 do Write ('*');
end.
```

```
{1.7}
program One7T92;
{ -- This program will display a 4-line statement for ISOP. }
  var
    Name, Title, Group: String[25];

begin
  Write ('Enter name: ');  Readln (Name);
  Write ('Enter title: '); Readln (Title);
  Write ('Enter group: '); Readln (Group);
  Writeln (Name, ' IS A ', Title, ' WITHIN THE');
  Writeln (Group, ' GROUP AND');
  Writeln ('HAS BEEN SELECTED TO PARTICIPATE IN');
  Writeln ('THE ISOP.');
end.


{1.8}
program One8T92;
{ -- This program will display a dollar sign next to an amount. }
  var
    St:     String[7];
    Code:   Integer;
    Amount: Real;

begin
  Write ('Enter amount: ');  Readln (St);
  Val(St, Amount, Code);
  if Amount >= 2000 then
    Writeln ('$2000.00')
  else
    Writeln ('$', St);
end.


{1.9}
program One9T92;
{ -- This program will display an acronym for business words. }
  var
    St: String[80];
    I:  Byte;

begin
  Write ('Enter words: ');  Readln (St);
  Write (Copy(St, 1, 1));
  for I := 2 to Length(St) - 1 do
    if Copy(St, I , 1) = ' ' then
      Write (Copy(St, I + 1, 1));
end.
```

```
{1.10}
program One10T92;
{ -- This program will calculate QUALITY hours and minutes. }
  var
    N, M, Total, Hours, Min: LongInt;

begin
  Write ('Enter number of technicians, N: '); Readln (N);
  Write ('Enter number of minutes, M: ');     Readln (M);
  Total := 50 * 5 * N * M;
  Hours := Total div 60;
  Min   := Total - Hours * 60;
  Writeln (Hours, ' HOURS ', Min, ' MINUTES');
end.
```

```
{2.1}
program Two1T92;
{ -- This program will display a speech indented. }
  var
    Line: Array [1..10] of String[40];
    I, J: Byte;
    Ch:   Char;

begin
  I := 0;
  repeat
    Inc(I);
    Write ('Enter Line: ');  Readln (Line[I]);
  until Line[I] = '';
  for J := 1 to I - 1 do begin
    Ch := Line[J,1];
    if Ch = 'I' then
      Writeln (Line[J])
    else if Ch in ['A' .. 'H'] then
      Writeln (' ': 4, Line[J])
    else
      Writeln (' ': 8, Line[J]);
  end;
end.


{2.2}
program Two2T92;
{ -- This program will display a number in words. }
  const
    Words: Array[1..27] of String[10] =
        ('ONE', 'TWO', 'THREE', 'FOUR', 'FIVE', 'SIX', 'SEVEN',
         'EIGHT', 'NINE', 'TEN', 'ELEVEN', 'TWELVE', 'THIRTEEN',
         'FOURTEEN', 'FIFTEEN', 'SIXTEEN', 'SEVENTEEN',
         'EIGHTEEN', 'NINETEEN', 'TWENTY', 'THIRTY', 'FOURTY',
         'FIFTY', 'SIXTY', 'SEVENTY', 'EIGHTY', 'NINETY');
  var
    Num, Units, Tens: Byte;

begin
  Write ('Enter number: ');  Readln (Num);
  if Num < 20 then
    Writeln (Words[Num])
  else begin
    Tens  := Num div 10;
    Units := Num - Tens * 10;
    Write (Words[18 + Tens]);
    if Units > 0 then
      Write ('-', Words[Units]);
  end;
end.
```

```
{2.3}
program Two3T92;
{ -- This program will display selected items from a NRD menu. }
uses Crt;
  const
    Crit: Array [1..7] of String[50] =
      ('DEMONSTRATED INTEREST IN INFORMATION MANAGEMENT.',
       'DEMONSTRATED LEADERSHIP SKILLS.',
       'STRONG GPA/PERFORMANCE HISTORY.',
       'AT LEAST TWO COURSES IN ANY PROGRAMMING LANGUAGE.',
       'INTERNSHIP OR WORK EXPERIENCE.',
       'EFFECTIVE ORAL AND WRITTEN COMMUNICATION SKILLS.',
       'CAREER DEVELOPMENT POTENTIAL.');
  var
    Name, Degree, Items: String[40];
    I, Num: Byte;
    Ist:    String[1];

begin
  Write ('Enter name: ');    Readln (Name);
  Write ('Enter degree: ');  Readln (Degree);
  for I := 1 to 7 do Writeln (I, '. ', Crit[I]);
  Writeln;
  Write ('Select up to 7 items: ');  Readln (Items);

  ClrScr;
  Writeln (Name);  Writeln (Degree);
  Num := 0;
  for I := 1 to 7 do begin
    Str(I, Ist);
    if Pos(Ist,Items) > 0 then begin
      Inc(Num);
      Writeln;  Writeln (Num, '. ', Crit[I]);
    end;
  end;
end.
```

```
{2.4}
program Two4T92;
{ -- This program will rate a speech. }
  const
    Cat: Array [1..7] of String[16] =
        ('SPEECH VALUE', 'PREPARATION', 'MANNER', 'ORGANIZATION',
         'OPENING', 'BODY OF SPEECH', 'CONCLUSION');
    Verbal: Array [1..5] of String[15] =
        ('EXCELLENT', 'ABOVE AVERAGE', 'SATISFACTORY',
         'SHOULD IMPROVE', 'MUST IMPROVE');
  var
    Rating:        Array [1..7] of String[15];
    I, Num, Total: Byte;
    Ave:           Real;

begin
  for I := 1 to 7 do begin
    Write ('Enter rating for ', Cat[I], ': ');
    Readln (Rating[I]);
  end;
  Total := 0;
  for I := 1 to 7 do begin
    Num := 1;
    while (Rating[I] <> Verbal[Num]) and (Num < 7) do
      Inc(Num);
    Writeln (Cat[I], ': ', Num);
    Inc(Total, Num);
  end;
  Writeln;
  Ave := Total / 7.0;
  Writeln ('AVERAGE NUMERICAL RATING = ', Ave: 2:1);
  Writeln ('SPEECH RATING = ', Verbal[Round(Ave)]);
end.
```

```
{2.5}
program Two5T92;
{ -- This program will format GTEDS MISSION statement. }
  const
    St: Array [1..4] of String[55] =
      ('BE THE CUSTOMER-ORIENTED LEADER AND PROVIDER-OF-CHOICE ',
       'OF QUALITY INFORMATION PRODUCTS AND SERVICES IN THE ',
       'TELECOMMUNICATIONS MARKETPLACE AND SELECTED OTHER ',
       'RELATED MARKETS IN SUPPORT OF GTE''S TELOPS GOALS.');
   var
     State: String[220];
     Line: String[40];
     Word: String[20];
     Ch:   Char;
     NumCh, I, J, N: Byte;

begin
  Write ('Enter N: ');  Readln (N);
  State := St[1] + St[2] + St[3] + St[4];
  Word  := '';  Line := '';
  for I := 1 to Length(State) do begin
    Ch := State[I];
    Word := Word + Ch;
    if Ch in [' ', '-', '.'] then begin
      NumCh := Length(Line) + Length(Word);
      if Ch = ' ' then Dec(NumCh);
      if NumCh > N then
        begin
          Writeln (Line);
          Line := Word;
        end
      else
        Line := Line + Word;
      Word := '';
    end;
  end;  { -- for I }
  Writeln (Line + Word);
end.
```

```
{2.6}
program Two6T92;
{ -- This program will change (.) to (?) at end of sentence. }
  const
    Quest: Array[1..5] of String[5] =
         ('WHAT', 'WHY', 'HOW', 'WHO', 'WHERE');
  var
    Par:       String[255];
    FirstW:    String[20];
    FirstWord: Boolean;
    Ch:        Char;
    I, J:      Byte;

begin
  Write ('Enter paragraph: ');  Readln (Par);  Writeln;
  FirstW := '';  FirstWord := True;
  for I := 1 to Length(Par) do begin
    Ch := Par[I];
    if (Ch = ' ') and (Length(FirstW) > 0) then
      FirstWord := False
    else if Ch in ['.', '!', '?'] then
      begin
        if Ch = '.' then
          for J := 1 to 5 do
            if FirstW = Quest[J] then Ch := '?';
        FirstW := '';  FirstWord := True;
      end
    else if FirstWord and (Ch <> ' ') then
      FirstW := FirstW + Ch;
    Write(Ch);
  end;
end.
```

```
{2.7}
program Two7T92;
{ -- This program will print names in the office at a beep. }
  const
    Name: Array[1..14] of String[10] = ('DAVID', 'DON', 'DOUG',
         'GRANDVILLE', 'JAMES', 'JIM', 'JOHN', 'LINDA', 'MARIE',
         'MATT', 'PAULA', 'ROBERT', 'SHELLEY', 'TOM');
    Start: Array[1..14] of Integer = (0700, 0800, 0730, 1230,
          1130, 0900, 0700, 1230, 0700, 1230, 0700, 0800,
          0630, 1100);
    Quit: Array[1..14] of Integer = (1600, 1700, 1630, 2100,
          2200, 1800, 1600, 2300, 1600, 2300, 1600, 1700,
          1530, 1930);
   var
     Time, I, Num: Integer;
     Day:          String[10];
     InOffice:     Boolean;

begin
  Write ('Enter time: ');  Readln (Time);
  Write ('Enter day: ');   Readln (Day);
  Num := 0;
  for I := 1 to 14 do begin
    if (Start[I] <= Time) and (Time <= Quit[I]) then
      if (Day <> 'SUNDAY') and (Day <> 'SATURDAY') then begin
        InOffice := True;
        if (Name[I] = 'JAMES') and (Day = 'MONDAY') then
          InOffice := False;
        if (Name[I] = 'LINDA') and (Day = 'FRIDAY') then
          InOffice := False;
        if (Name[I] = 'MATT')  and (Day = 'MONDAY') then
          InOffice := False;
        if InOffice then begin
          Inc(Num);
          if Num = 1 then
            Write (Name[I])
          else
            Write (', ', Name[I]);
        end;
      end;
  end;  { -- for I }
  if Num = 0 then Writeln ('NONE');
end.
```

```
{2.8}
program Two8T92;
{ -- This program will randomly assign titles to a team. }
  const
    Name: Array[1..7] of String[7] = ('WILL', 'DARLENE',
          'JEFF', 'LIZ', 'LORI', 'MARY', 'PING');
    Title: Array[1..5] of String[9] = ('AUTHOR',
          'MODERATOR', 'READER', 'RECORDER', 'INSPECTOR');
  var
    I, J, X: Byte;
    TName:   Array[1..5] of String[7];
    Valid:   Boolean;

begin
  Randomize;
  Write ('Enter author''s name: ');  Readln (TName[1]);

  { -- Choose a moderator: TName[2] }
  if TName[1] = Name[1] then
    TName[2] := Name[2]
  else if TName[1] = Name[2] then
    TName[2] := Name[1]
  else
    TName[2] := Name[Random(2) + 1];

  { -- Choose next 3 title names. }
  for I := 3 to 5 do begin
    repeat
      Valid := True;
      X := Random(7) + 1;
      for J := 1 to I do
        if Name[X] = TName[J] then Valid := False;
    until Valid;
    TName[I] := Name[X];
  end;
  { -- Display all 5 titles and names. }
  for I := 1 to 5 do
    Writeln (Title[I], ' - ', TName[I]);
end.
```

```
{2.9}
program Two9T92;
{ -- This program will sort a list of names with area codes. }
  var
    Name: Array[1..15] of String[15];
    I, J, A, Area1, Area2, Num: Integer;
    X: String[15];

begin
  Write ('Enter two area codes: ');  Readln (Area1, Area2);
  Write ('Enter number of names: '); Readln (Num);
  for I := 1 to Num do begin
    Write ('Enter name: ');  Readln (Name[I]);
  end;
  for I := 1 to Num - 1 do
    for J := I + 1 to Num do
      if Name[I] > Name[J] then begin
        X := Name[I];  Name[I] := Name[J];  Name[J] := X;
      end;

  if Area1 > Area2 then begin
    A := Area1;  Area1 := Area2;  Area2 := A;
  end;

  for I := 1 to (Num + 1) div 2 do
    Writeln (Area1, ' - ', Name[I]);
  for I := (Num + 1) div 2 + 1 to Num do
    Writeln (Area2, ' - ', Name[I]);
end.



{2.10}
program Two10T92;
{ -- This program will adjust a golf score by handicap. }
  const
    Par: Array[1..9] of Byte = (5,4,4,4,3,4,4,3,5);
  var
    G, A: Array[1..9] of Byte;
    Bog:  Array[1..3] of Byte;
    Hand, RHand, I, B, ParTot:    Byte;
    GTot, ATot, Sing, Doub, Trip: Byte;
    Diff:                         Integer;
    Adjusted:                     Boolean;

begin
  ParTot := 0;  GTot := 0;  ATot := 0;
  Sing   := 0;  Doub := 0;  Trip := 0;
  Write ('Enter handicap: ');  Readln (Hand);
  Write ('Enter gross scores: ');
  Readln (G[1],G[2],G[3],G[4],G[5],G[6],G[7],G[8],G[9]);
  Write ('HOLE #:');
  for I := 1 to 9 do Write (I: 4);
  Writeln;
  Write ('PAR:   ');
```

```
  for I := 1 to 9 do begin
    Write (Par[I]: 4);
    ParTot := ParTot + Par[I];
  end;
  Writeln;
  Write ('GROSS: ');
  for I := 1 to 9 do begin
    Write (G[I]: 4);
    GTot := GTot + G[I];
  end;
  Writeln;
  Write ('ADJUST:');

  { -- Determine # of tripple and double bogeys allowed. }
  if Hand > 9 then begin
    Bog[3] := Hand - 9;
    Bog[2] := 9 - Bog[3];
    Bog[1] := 0
    end
  else begin
    Bog[3] := 0;
    Bog[2] := Hand;
    Bog[1] := 9 - Bog[2];
  end;

  { -- Adjust the gross scores by Handicap. }
  for I := 1 to 9 do begin
    Diff := G[I] - Par[I];
    Adjusted := False;
    B := 3;
    while not Adjusted and (B > 0) do begin
      if (Bog[B] > 0) and (Diff >= B) then begin
        A[I] := Par[I] + B;
        Dec(Bog[B]);
        Adjusted := True;
      end;
      Dec(B);
    end;
    if not Adjusted then
      A[I] := G[I];
  end;  { -- for I }

  { -- Display the adjusted scores and totals. }
  for I := 1 to 9 do begin
    Write (A[I]: 4);
    ATot := ATot + A[I];
  end;
  Writeln;  Writeln;
  Writeln ('PAR TOTAL: ', ParTot);
  Writeln ('GROSS TOTAL: ', GTot);
  Writeln ('ADJUST TOTAL: ', ATot);
  Writeln ('ROUND HANDICAP: ', ATot - ParTot);
end.
```

```
{3.1}
program Thr1T92;
{ -- This program will move a triangle of GTEDS around screen. }
uses Crt;
  const
    A: Array[1..7] of String[11] =
       ('           ',
        '     G     ',
        '    T T    ',
        '   E   E   ',
        '  D     D  ',
        ' SDETGTEDS ',
        '           ');
   var
     Row, Col, I: Integer;
     Ch: Char;

begin
  ClrScr;
  Row := 9;  Col := 34;
  repeat
    for I := 1 to 7 do begin
      GotoXY (Col, Row + I);
      Writeln (A[I]);
    end;
    for I := 1 to 1000 do
      if KeyPressed then Ch := ReadKey;
    case Upcase(CH) of
      'I' : Dec(Row);
      'M' : Inc(Row);
      'J' : Dec(Col);
      'K' : Inc(Col);
    end;
    if Row = 0 then begin
      Row := 1;  Ch := ' ';  end;
    if Col = 0 then begin
      Col := 1;  Ch := ' ';  end;
    if Row = 18 then begin
      Row := 17; Ch := ' ';  end;
    if Col = 69 then begin
      Col := 68; Ch := ' ';  end;
  until Ch = Chr(27);
  Ch := ' ';
end.
```

```
{3.2}
program Thr2T92;
{ -- This program will display a date in 1992 after # of days. }
  const
    Day:   Array[1..6] of String[10] = ('TUESDAY', 'WEDNESDAY',
           'THURSDAY', 'FRIDAY', 'SATURDAY', 'MONDAY');
    Month: Array[1..12] of Integer =
           (31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    MName: Array[1..12] of String[10] = ('JANUARY', 'FEBRUARY',
           'MARCH', 'APRIL', 'MAY', 'JUNE', 'JULY', 'AUGUST',
           'SEPTEMBER', 'OCTOBER', 'NOVEMBER', 'DECEMBER');
  var
    X, D, I, Sum: Integer;

begin
  Write ('Enter X: ');  Readln (X);
  Inc(X);
  D := (X MOD 6) + 1;
  Write(Day[D], ' ');
  X := X + (X + 1) div 6;
  Sum := 0;  I := 1;
  while Sum + Month[I] < X do begin
    Sum := Sum + Month[I];  Inc(I);
  end;
  Writeln (MName[I], ' ', X - Sum);
  if Day[D] = 'SATURDAY' then begin
    Inc(X);
    while Sum + Month[I] < X do begin
      Sum := Sum + Month[I];  Inc(I);
    end;
    Writeln ('SUNDAY ', MName[I], ' ', X - Sum);
  end;
end.



{3.3}
program Thr3T92;
{ -- This program will release program modules for PWS. }
  var
    Name, Prog:   Array [1..9] of String[10];
    Comp, Rel:    Array [1..9] of String[1];
    St, Module:   String[20];
    I, J, L, Num: Byte;
    AllDone, ModComp, ModRel: Boolean;

begin
  I := 0;  Num := 0;  AllDone := False;  ModRel := False;
  repeat
    I := Num + 1;
    Write ('Enter name, program: ');  Readln (St);
      L := Length(St);
      Name[I] := Copy(St, 1, L - 5);
      Prog[I] := Copy(St, L - 3, 4);
```

```
      { -- Find previous Name/Prog or make addition. }
      J := 1;
      while (J < I) and ((Name[J] <> Name[I]) or
      (Prog[J] <> Prog[I])) do
        Inc(J);
      I := J;
      if I > Num then Num := I;

    Write ('Enter completed, release: ');  Readln (St);
      Comp[I] := Copy(St, 1, 1);
      Rel[I]  := Copy(St, 3, 1);
      if Rel[I] = 'Y' then Comp[I] := 'Y';
      ModComp := (Comp[I] = 'Y');

    { -- Check if Module completed by all at least 1 released. }
    if ModComp then begin
      ModRel := False;
      for J := 1 to Num do
        if (Prog[J] = Prog[I]) then begin
          if (Comp[J] <> 'Y') then
            ModComp := False;
          if (Rel[J] = 'Y') then
            ModRel := True;
        end;

      { -- If Module completed by all and 1 or more released. }
      if ModComp and ModRel then begin
        Writeln ('MODULE ', Prog[I], ' HAS BEEN RELEASED');
        Module := Prog[I];
        for J := 1 to Num do
          if Prog[J] = Module then Prog[J] := '';
        AllDone := True;
        for J := 1 to Num do
          if Prog[J] <> '' then AllDone := False;
      end;
    end;  { -- If ModComp }
  until AllDone;
end.
```

```
{3.4}
program Thr4T92;
{ -- This program will produce acronyms for phone numbers. }
  const
    B: Array [1..18] of String[5] = ('AGENT', 'SOAP', 'MONEY',
    'JEWEL', 'BALL', 'LOANS', 'CARE', 'SAVE', 'CALL', 'PAVE',
    'KEEP', 'KINGS', 'KNIFE', 'KNOCK', 'JOINT', 'JUICE',
    'LOBBY', 'RATE');
    L1: String[9] = ' ADGJMPTW';
    L2: String[9] = ' BEHKNRUX';
    L3: String[9] = ' CFILOSVY';
  var
    I, J, K, L: Byte;
    Ph, Num:    String[8];
    P4, P5, X:  String[5];
    C:          String[1];
    A:          Array[1..18] of String[5];

begin
  { -- Sort the data alphabetically. }
  for I := 1 to 18 do A[I] := B[I];
  for I := 1 to 17 do
    for J := I + 1 to 18 do
      if A[I] > A[J] then begin
        X := A[I];  A[I] := A[J];  A[J] := X;
      end;

  Write ('Enter phone #: ');  Readln (Ph);
  P4 := Copy(Ph, 5, 4);  P5 := Copy(Ph, 3, 1) + P4;
  { -- Convert words to number strings }
  for I := 1 to 18 do begin
    L := Length(A[I]);  Num := '';
    for J := 1 to L do begin
      K := 2;  C := Copy(A[I], J, 1);;
      while (L1[K] <> C) and (L2[K] <> C) and (L3[K] <> C) do
        Inc(K);
      Num := Num + Chr(48 + K);
    end;
    if (L = 4) and (Num = P4) then
      Writeln (Copy(Ph, 1, 4), A[I])
    else if (L=5) and (Num = P5) then begin
      Write (Copy(Ph, 1, 2),  Copy(A[I], 1, 1), '-');
      Writeln (Copy(A[I], L - 3, 4));
    end;
  end;
end.
```

```
{3.5}
program Thr5T92;
{ -- This program will find seven 7-digit squares in base 8. }
  var
    Num1V, Num2, Power, Num: LongInt;
    I, J, K, X, Digit, SNum: Integer;
    Num1, NumSt:             String[4];
    Square:                  String[7];
    Dup:                     Array[0..7] of Boolean;
    Valid:                   Boolean;

begin
  Num := 1242;  SNum := 0;
  repeat
    Str (Num, Num1);

    { -- Convert Num1 to base 10 number Num1V }
    Num1V := 0;
    for I := 1 to 4 do begin
      Digit := Ord(Num1[I]) - Ord('0');
      Power := 1;
      for J := 1 to Length(Num1) - I do
        Power := Power * 8;
      Num1V := Num1V + Digit * Power;
    end;

    Num1V := Num1V * Num1V;

    Square := '';  Valid := True;
    for I := 0 to 7 do
      Dup[I] := False;

    { -- Convert Num1V to Base8 number }
    J := Trunc(Ln(Num1V) / Ln(8));
    repeat
      Power := 1;
      for K := 1 to J do  Power := Power * 8;
      X := Trunc(Num1V / Power);
      { -- Check for duplicate digits. }
      if not Dup[X] then begin
        Dup[X] := True;
        Square := Square + Chr(48 + X);
        Num1V := Num1V - X * Power;
        end
      else
        Valid := False;
      Dec(J);
    until (J < 0) or not Valid;

    if Valid then begin
      Inc(SNum);
      Writeln (Square, '  ', Num);
    end;
```

```
    { -- increment to next base 8 number }
    repeat
      Inc(Num);
      Str(Num, NumSt)
    until (Pos('8',NumSt) = 0) and (Pos('9',NumSt) = 0);

  until SNum = 7;
end.



{3.6}
program Thr6T92;
{ -- This program will find 3 distinct integers that are pairwise
  -- relatively prime such that they sum to N. }
var
  X, Y, Z, N, I: Integer;
  Found:         Boolean;

begin
  Write ('Enter N: ');  Readln (N);
  X := 2 + (N mod 2);  Found := False;
  while (X < N div 3) and not Found do begin
    Y := X + 1;
    while (Y < (N - X) div 2) and not Found do begin
      Z := N - X - Y;  Found := True;
      for I := 2 to Y do
        if ((X mod I = 0) and (Y mod I = 0)) or
           ((X mod I = 0) and (Z mod I = 0)) or
           ((Y mod I = 0) and (Z mod I = 0)) then
          Found := False;
      if Found then
        Writeln (X, ' + ', Y, ' + ', Z, ' = ', N)
      else
        Inc(Y);
    end;
    Inc(Z);
  end;
end.
```

```
{3.7}
program Thr7T92;
{ -- This program will print combinations of 6 soccer players. }
uses Crt;
  var
    A:    Array [1..9] of Integer;
    Name: Array [1..9] of String[10];
    X:    String[10];
    Ch:   Char;
    I, J, M, L, N, S, Sub: Byte;

begin
  Name[1] := 'ANDY';  Name[2] := 'DAN';  Name[3] := 'DOUG';
  Name[4] := 'JACK';  Name[5] := 'MIKE'; Name[6] := 'YEHIA';

  Write ('Enter number of substitutes: ');  Readln (Sub);
  L := 6 + Sub;
  for I := 7 to L do begin
    Write ('Enter name: ');  Readln (Name[I]);
  end;
  { -- Sort names with substitutes. }
  for I := 1 to L - 1 do
    for J := I + 1 to L do
      if Name[I] > Name[J] then begin
        X := Name[I];  Name[I] := Name[J];  Name[J] := X;
      end;

  M := 6;
  for I := 1 to M do   A[I] := M - I + 1;
  N := 1;  A[1] := A[1] - 1;  S := 0;
  while N <= M do begin
    A[N] := A[N] + 1;
    if N > 1 then
      for I := N-1 downto 1 do A[I] := A[I+1] + 1;
    if A[N] <= L - N + 1 then begin
      Inc(S);
      Write (S, ' ',Name[A[M]]);
      for I := M - 1 downto 1 do Write (',',Name[A[I]]);
      Writeln;
      N := 0;
      if S mod 24 = 0 then Ch := ReadKey;
    end;
    Inc(N);
  end;
end.
```

```
{3.8}
program Thr8T92;
{ -- This program displays the Bill Date and the Due Date. }
{ -- January 1, 1992 was a Wednesday. }
  const
    Mname: Array [1..12] of String[9] = ('JANUARY', 'FEBRUARY',
           'MARCH', 'APRIL', 'MAY', 'JUNE', 'JULY,', 'AUGUST',
           'SEPTEMBER', 'OCTOBER', 'NOVEMBER', 'DECEMBER');
    Mon:   Array [1..12] of Integer =
           (31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
    Dname: Array [1..7] of String[9] = ('TUESDAY', 'WEDNESDAY',
           'THURSDAY', 'FRIDAY', 'SATURDAY', 'SUNDAY', 'MONDAY');
  var
    I, T, H, Hol, Wkend, X, MNum, Cycle, NumDays: Integer;
    Mhol, Dhol: Array [1..12] of Integer;
    Day, Days:  Array [1..2]  of Integer;

begin
  Write ('Enter month of bill: ');  Readln (MNum);
  Write ('Enter cycle number: ');   Readln (Cycle);
  Write ('Enter number of days: '); Readln (NumDays);
  H := 1;
  Write ('Enter holiday MM, DD: ');  Readln (Mhol[H], Dhol[H]);
  while Mhol[H] > 0 do begin
    H := H + 1;
    Write ('Enter holiday MM, DD: ');  Readln (Mhol[H], Dhol[H]);
  end;
  Dec(H);  Writeln;
  Days[1] := 0;
  for I := 1 to MNum - 1 do
    Days[1] := Days[1] + Mon[I];
  Day[1] := 3 * Cycle - 2;  Day[2] := Day[1] + NumDays;
  Days[2] := Days[1] + Day[2];
  Days[1] := Days[1] + Day[1];
  for T := 1 to 2 do begin
    Hol := 1;  Wkend := 1;
    { -- Decrement days counter if holiday or weekend. }
    while (Hol = 1) or (Wkend = 1) do begin
      Hol := 0;  Wkend := 0;
      if Day[T] > Mon[MNum] then begin
        Day[T] := Day[T] - Mon[MNum];
        Inc(MNum)
      end;
      for I := 1 to H do
        if (Mhol[I] = MNum) and (Dhol[I] = Day[T]) then begin
          Inc(Day[T]);
          Inc(Days[T]);  Hol := 1;
        end;
      X := Days[T] mod 7;
      if (X = 4) or (X = 5) then begin  { -- Sat. or Sun. }
        Inc(Day[T]);
        Inc(Days[T]);  Wkend := 1;
      end;
    end;  { -- while }
    if T = 1 then Write ('BILL ') else Write ('DUE ');
```

```
    Write   ('DATE: ', Dname[X+1], ' ', Mname[MNum], ' ');
    Writeln (Day[T]);
  end;  { -- for T }
end.



{3.9}
program Thr9T92;
{ -- This program will calculate the area of a polygon room. }
  var
    I, L, Sides, Code, Sum, Area: Integer;
    Mov:  String[3];
    Dir:  Array[1..10] of String[1];
    Dist: Array[1..10] of Integer;

begin
  Write ('Enter number of sides: ');  Readln (Sides);
  for I := 1 to Sides do begin
    Write ('Enter movement: ');  Readln (Mov);
    Dir[I] := Copy(Mov, 1, 1);
    L := Length(Mov);
    Mov := Copy(Mov, 2, L - 1);
    Val(Mov, Dist[I], Code);
    { -- Subtract Down and Left directions }
    if (Dir[I] = 'D') or (Dir[I] = 'L') then
      Dist[I] := -Dist[I];
  end;
  { -- Multiply length by width to obtain rectangle area, }
  { -- then add or subtract area from overall area. }
  I := 1;  Sum := 0;  Area := 0;
  repeat
    Sum := Sum + Dist[I];
    Area := Area + (Sum * Dist[I+1]);
    Inc(I, 2);
  until (I > Sides);
  Writeln ('AREA = ', Abs(Area), ' SQUARE FEET');
end.
```

```
{3.10}
program Thr10T92;
{ -- This program will display the reasons a Rubik's Cube is }
{ -- unsolvable. }
  const
   Side:  Array[1..6] of String[7] =
          ('TOP:   ', 'FRONT: ', 'RIGHT: ', 'BACK:  ',
           'LEFT:  ', 'BOTTOM:');
   EdgeS: String[60] =
   'T2P2 T6R2 T8F2 T4L2 F4L6 F6R4 R6P4 P6L4 F8B2 R8B6 P8B8 L8B4';
  var
    Col:       Array[1..6, 1..9] of String[1];
    I, J, K:   Byte;
    MidUnique: Boolean;
    Colors:    String[17];
    S1, S2, N1, N2, ENum: Byte;


begin
  for I := 1 to 6 do begin
    Write ('Enter colors on ', Side[I], ' ');  Readln (Colors);
    for J := 1 to 9 do
      Col[I,J] := Copy(Colors, J * 2 - 1, 1);
  end;

   MidUnique := True;
   for I := 1 to 5 do
     for J := I + 1 to 6 do
       if Col[I, 5] = Col[J, 5] then
         MidUnique := False;

   if not MidUnique then
     Writeln ('COLORS ON MIDDLE SQUARES ARE NOT UNIQUE');

   ENum := 0;
   for K := 1 to 12 do begin
     S1 := Pos(EdgeS[K*5 - 4], 'TFRPLB');
     N1 := Ord(EdgeS[K*5 - 3]) - 48;
     S2 := Pos(EdgeS[K*5 - 2], 'TFRPLB');
     N2 := Ord(EdgeS[K*5 - 1]) - 48;
     if Col[S1, N1] = Col[S2, N2] then
       Inc(ENum);
   end;

   Writeln ('NUMBER OF EDGE PIECES HAVING SAME COLOR: ', ENum);
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '93 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T93;
{ -- This program displays six lines with "GTEDS". }
{ -- The solution could also be done with 6 Writeln statements. }
  var
    I, J: Byte;

begin
  for I := 1 to 6 do begin
    for J := 1 to 7 - I do begin
      Write ('GTEDS', ' ':I);
    end;
    Writeln;
  end;
end.



{1.2}
program One2T93;
{ -- This program displays the number of programmers placed. }
  var
    N, M: Integer;

begin
  Write ('Enter N: ');  Readln (N);
  Write ('Enter M: ');  Readln (M);
  Writeln (N * 15 - M, ' PROGRAMMERS');
end.



{1.3}
program One3T93;
{ -- This program will format the number N million with commas. }
  var
    N:   Real;
    NSt: String[12];

begin
  Write ('Enter N: ');  Readln (N);
  STR (N * 1E6 :9:0, NSt);
  Insert (',', NSt, 7);  Insert (',', NSt, 4);
  Writeln (NSt, ' ACCESS LINES');
end.
```

```
{1.4}
program One4T93;
{ -- This program will total # of students on 5 USF campuses. }
  const
    Campus: Array[1..5] of String[14] =  ('Tampa',
     'St. Petersburg', 'Fort Myers', 'Lakeland', 'Sarasota');
  var
    Num, Total: LongInt;
    I:          Byte;

begin
  Total := 0;
  for I := 1 to 5 do begin
    Write ('Enter # at ', Campus[I], ': ');  Readln (Num);
    Total := Total + Num;
  end;
  Write (Total, ' STUDENTS');
end.


{1.5}
program One5T93;
{ -- This program will determine if person qualifies for ISOP. }
  var
    Name:   String[12];
    Level:  Byte;
    Desire: String[3];

begin
  Write ('Enter Name: ');    Readln (Name);
  Write ('Enter level: ');   Readln (Level);
  Write ('Enter desire: ');  Readln (Desire);
  Write (Name, ' IS ');
  if (level < 5) or (Desire = 'NO') then
    Write ('NOT ');
  Writeln ('A POSSIBLE CANDIDATE FOR ISOP');
end.
```

```
{1.6}
program One6T93;
{ -- This program will display preferred skills for curriculum. }
  var
    Curr: String[15];

begin
  Write ('Enter curriculum: ');  Readln (Curr);
  if Curr = 'MVS/COBOL' then
    begin
      Writeln ('COBOL');
      Writeln ('JCL');
      Writeln ('MVS/ESA');
      Writeln ('TSO/ISPF');
      Writeln ('VSAM');
      Writeln ('ANSI SQL');
      Writeln ('DB2');
      Writeln ('IMS');
    end
  else     { -- Curr = 'C/UNIX' }
    begin
      Writeln ('C');
      Writeln ('UNIX');
      Writeln ('ANSI SQL');
      Writeln ('OSF/MOTIF');
      Writeln ('SHELL PROGRAMMING');
    end;
end.


{1.7}
program One7T93;
{ -- This program will print the first N letters of alphabet. }
  var
    I, N: Byte;

begin
  Write ('Enter N: ');  Readln(N);
  for I := 1 to N do
    Write (Chr(64 + I));
end.
```

```
{1.8}
program One8T93;
{ -- This program will calculate the increase in salary. }
  var
    Salary, Increase: Real;
    Rating:           String[13];

begin
  Write ('Enter salary: ');  Readln (Salary);
  Write ('Enter rating: ');  Readln (Rating);
  if Rating = 'EXCELLENT' then
    Increase := Salary * 0.10
  else if Rating = 'ABOVE AVERAGE' then
    Increase := Salary * 0.07
  else if Rating = 'GOOD' then
    Increase := Salary * 0.05
  else
    Increase := 0.0;
  Writeln ('NEW SALARY = $', Salary + Increase: 7:2);
end.



{1.9}
program One9T93;
{ -- This program will display a Service Order. }
  var
    SO: String[7];
    Ch: Char;

begin
  Write ('Enter order: ');  Readln (SO);
  Ch := SO[1];
  if Length(SO) > 1 then
    Writeln (Ch)
  else
    Case Ch of
      'I':  Writeln ('INSTALL');
      'C':  Writeln ('CHANGE');
      'R':  Writeln ('RECORDS');
      'O':  Writeln ('OUT');
      'F':  Writeln ('FROM');
      'T':  Writeln ('TO');
    end;
end.
```

```
{1.10}
program One10T93;
{ -- This program will compute a GPA for 5 classes. }
  var
    G:           Char;
    I, Num, Sum: Byte;

begin
  Sum := 0;  Num := 5;
  for I := 1 to 5 do begin
    Write ('Enter grade: ');  Readln (G);
    case G of
      'A': Sum := Sum + 4;
      'B': Sum := Sum + 3;
      'C': Sum := Sum + 2;
      'D': Sum := Sum + 1;
    end;
    if G = 'W' then Num := Num - 1;
  end;
  Writeln ('GPA = ', Sum / Num : 4:3);
end.
```

```
{2.1}
program Two1T93;
{ -- This program will randomly generate #s between X and Y. }
  var
    I, N, X, Y, Min, Max: ShortInt;

begin
  Randomize;
  Write ('Enter N: ');  Readln (N);
  Write ('Enter X, Y: ');  Readln (X, Y);
  if X < Y then begin
    Min := X;  Max := Y;  end
  else begin
    Min := Y;  Max := X;
  end;

  for I := 1 to N do
    Write (Random(Max - Min + 1) + Min, ' ');

end.
```

```
{2.2}
program Two2T93;
{ -- This program will sort names according to their title. }
  const
    Titles: Array[1..7] of String[4] =
      ('P', 'PA', 'SA', 'SE', 'SSE', 'ASE', 'SASE');
  var
    Name:       Array [1..10] of String[20];
    Level:      Array [1..10] of Integer;
    Title:      String[4];
    TempN:      String[12];
    I, J, N, T: Byte;

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter name: ');  Readln (Name[I]);
    Write ('Enter title: '); Readln (Title);
    Name[I] := Name[I] + ' - ' + Title;
    J := 1;
    while Titles[J] <> Title do J := J + 1;
    Level[I] := J;
  end;

  for I := 1 to N - 1 do
    for J := I + 1 to N do
      if (Level[I] < Level[J])
      or ((Level[I] = Level[J]) and (Name[I] > Name[J])) then
       begin
        TempN := Name[I]; Name[I] := Name[J];   Name[J] := TempN;
        T := Level[I];   Level[I] := Level[J]; Level[J] := T;
       end;

  for I := 1 to N do
    Writeln (Name[I]);
end.
```

```
{2.3}
program Two3T93;
{ -- This program will format a COBOL declaration. }
  var
    Field:            Array[1..15] of String[30];
    Level, PrevLevel: String[2];
    I, J, Inc:        Integer;

begin
  I := 0;
  repeat
    I := I + 1;
    Write ('Enter field: ');  Readln (Field[I]);
  until Field[I] = '';

  for J := 1 to I - 1 do begin
    Level := Copy(Field[J], 1, 2);
    if Level = '01' then
      Inc := 0
    else if Level > PrevLevel then
      Inc := Inc + 4
    else if Level < PrevLevel then
      Inc := Inc - 4;

    if Inc > 0 then
      Write (' ': Inc);
    Writeln (Field[J]);

    PrevLevel := Level;
  end;  { -- for J }
end.
```

```
{2.4}
program Two4T93;
{ -- This program will translate a word and calculate blocks. }
  var
    Word, Number:          String[30];
    I, Num, Blocks, Code: Integer;
    Digit, LastDigit:     Byte;
    NumSt:                 String[2];

begin
  Write ('Enter word: ');  Readln (Word);

  Number := '';
  for I := 1 to Length(Word) do begin
    Num := Ord(Word[I]) - Ord('A') + 1;
    Str(Num, NumSt);
    Number := Number + NumSt;
  end;
  Writeln ('NUMBER = ', Number);

  Blocks := 1;
  Val (Copy(Number,1,1), LastDigit, Code);
  for I := 2 to Length(Number) do begin
    Val (Copy(Number,I,1), Digit, Code);
    if Digit mod 2 <> LastDigit mod 2 then
      Blocks := Blocks + 1;
    LastDigit := Digit;
  end;

  Writeln ('BLOCKS = ', Blocks);
end.
```

```
{2.5}
program Two5T93;
{ -- This program will display N formatted telephone #s. }
  var
    Num:          Array[1..15] of String[10];
    Line:         String[4];
    I, N, Total: Byte;
    NPA, NXX, NextNPA, NextNXX: String[3];

begin
  Write ('Enter N: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter #: ');  Readln (Num[I]);
  end;
  Total := 1;  Num[I+1] := '     ';
  for I := 1 to N do begin
    NPA := Copy(Num[I], 1, 3);
    NXX := Copy(Num[I], 4, 3);
    Line:= Copy(Num[I], 7, 4);
    Write (NPA, '-', NXX, '-', Line);
    NextNPA := Copy(Num[I+1], 1, 3);
    NextNXX := Copy(Num[I+1], 4, 3);
    if (NPA <> NextNPA) then begin
      Writeln ('  TOTAL FOR NPA OF ', NPA, ' = ', Total);
      Writeln;
      Total := 1;
      end
    else begin
      Inc(Total);
      if NXX <> NextNXX then
        Writeln;
    end;
    Writeln;
  end;  { -- for I }
end.
```

```
{2.6}
program Two6T93;
{ -- This program will calculate product bought minus coupons. }
  var
    Prod, Coup:      Array[1..10] of String[1];
    Pric, Disc:      Array[1..10] of Real;
    Total, MaxDisc:  Real;
    I, J, NumProd, NumCoup, Ind: Byte;

begin
  I := 0;
  repeat
    Inc(I);
    Write ('Enter product: '); Readln (Prod[I]);
    if Prod[I] <> '9' then begin
      Write ('Enter price: ');  Readln (Pric[I]);
    end;
  until Prod[I] = '9';
  NumProd := I - 1;
  Writeln;

  J := 0;
  repeat
    Inc(J);
    Write ('Enter coupon: '); Readln (Coup[J]);
    if Coup[J] <> '9' then begin
      Write ('Enter discount: ');  Readln (Disc[J]);
    end;
  until Coup[J] = '9';
  NumCoup := J - 1;

  Total := 0;
  for I := 1 to NumProd do begin
    MaxDisc := 0;
    for J := 1 to NumCoup do
      if Prod[I] = Coup[J] then
        if Disc[J] > MaxDisc then begin
          MaxDisc := Disc[J];  Ind := J;
        end;
    Total := Total + Pric[I] - MaxDisc;
    Coup[Ind] := '*';
  end;

  Writeln;
  Writeln ('TOTAL = $', Total: 4:2);
end.
```

```
{2.7}
program Two7T93;
{ -- This program will display dates in other formats. }
  var
    Format: String[8];
    Date:   String[10];
    YYYY:   String[4];
    DD, MM: String[2];

begin
  Write ('Enter format: ');  Readln (Format);
  Write ('Enter date: ');    Readln (Date);
  if Format = 'ISO' then begin
    YYYY := Copy (Date, 1, 4);
    MM   := Copy (Date, 6, 2);
    DD   := Copy (Date, 9, 2);
    end
  else if Format = 'AMERICAN' then begin
    MM   := Copy (Date, 1, 2);
    DD   := Copy (Date, 4, 2);
    YYYY := Copy (Date, 7, 4);
    end
  else begin  { -- Format = 'EUROPEAN' }
    DD   := Copy (Date, 1, 2);
    MM   := Copy (Date, 4, 2);
    YYYY := Copy (Date, 7, 4);
  end;

  if Format <> 'ISO' then
    Writeln ('ISO = ', YYYY, '-', MM, '-', DD);
  if Format <> 'AMERICAN' then
    Writeln ('AMERICAN = ', MM, '-', DD, '-', YYYY);
  if Format <> 'EUROPEAN' then
    Writeln ('EUROPEAN = ', DD, '-', MM, '-', YYYY);
end.
```

```
{2.8}
program Two8T93;
{ -- This program will reverse the words in 1 or 2 sentences. }
  var
    Sent:      String;
    Word:      Array [1..10] of String[15];
    I, J, Num: Integer;
    Ch:        Char;

begin
  Write ('Enter sentence: ');  Readln (Sent);
  Num := 1;  Word[Num] := '';  I := 1;
  repeat
    Ch := Sent[I];
    if Ch = '.' then
      begin
        for J := Num downto 1 do
          if J = Num then
            Write (Word[J])
          else
            Write (' ', Word[J]);
        Write ('.  ');
        Num := 0;  Inc(I);
      end
    else
      if Ch <> ' ' then  { -- Add letter to word. }
        Word[Num] := Word[Num] + Ch
      else  { -- Word completed by a space. }
        begin
          Inc(Num);
          Word[Num] := '';
        end;
    Inc(I);
  until (I > Length(Sent));
end.
```

```
{2.9}
program Two9T93;
{ -- This program will print 4 smallest #s in a 4 x 4 matrix. }
  var
    I, J, K, X, Num: Byte;
    A: Array [1..4, 1..4] of ShortInt;
    B: Array [0..16] of ShortInt;
    OneDisplayed: Boolean;

begin
  for I := 1 to 4 do begin
    Write ('Enter row ', I, ': ');
    Readln (A[I,1], A[I,2], A[I,3], A[I,4]);
  end;

  for I := 1 to 4 do
    for J := 1 to 4 do
      B[(I - 1) * 4 + J] := A[I,J];

  for I := 1 to 15 do
    for J := I + 1 to 16 do
      if B[I] > B[J] then begin
        X := B[I];  B[I] := B[J];  B[J] := X;
      end;

  K := 1;  Num := 0;  B[0] := -99;
  repeat
    OneDisplayed := False;
    if B[K] <> B[K-1] then begin
      Writeln;
      Inc(Num);
      Write (Num, '. SMALLEST = ', B[K], ' OCCURS AT ');
      for I := 1 to 4 do
        for J := 1 to 4 do
          if B[K] = A[I,J] then begin
            if OneDisplayed then
              Write (', ')
            else
              OneDisplayed := True;
            Write ('(', I, ',', J, ')');
          end;
    end;  { -- if B[K] }
    Inc(K);
  until (Num = 4) and (B[K] <> B[K-1]);
end.
```

```
{2.10}
program Two10T93;
{ -- This program will print # of days between two dates. }
  const
    Month: Array [1..12] of Byte =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
  var
    M, D, Y, I, Days, Days2: Integer;

begin
  Write ('Enter month: ');  Readln (M);
  Write ('Enter day: ');    Readln (D);
  Write ('Enter year: ');   Readln (Y);
  Days := 0;  Days2 := 0;
  { -- October 25, 1967 }
  for I := 1 to 9 do
    Days2 := Days2 + Month[I];
  Days2 := Days2 + 25;

  for I := 1967 to Y - 1 do begin
    Days := Days + 365;
    if I mod 4 = 0 then Days := Days + 1;
  end;
  if (Y mod 4 = 0) and (M > 2) then Days := Days + 1;
  for I := 1 to M - 1 do
    Days := Days + Month[I];
  Days := Days + D;

  Writeln (Days - Days2, ' DAYS');
end.
```

```
{3.1}
program Thr1T93;
{ -- This program displays GTEDS squares relative to cursor. }
{ -- Cursor can be moved up, left, down, right: I, J, K, M. }
uses Crt;
  var
    R, C, X, A, B: Integer;
    K: Char;

begin
  ClrScr;  R := 5;  C := 5;  K := ' ';
  while not (K in ['1' .. '4']) do begin
    GotoXY (C, R);  Write ('#');  K := ' ';
    K := ReadKey;
    if K in ['I', 'J', 'K', 'M'] then begin
      GotoXY (C, R);  Write (' ');
      if K = 'I' then Dec(R);
      if K = 'M' then Inc(R);
      if K = 'J' then Dec(C);
      if K = 'K' then Inc(C);
    end;
  end;
  X := Ord(K) - Ord('0');
  if X = 1 then begin  A := 1;  B :=  0;  end;
  if X = 2 then begin  A := 1;  B := -1;  end;
  if X = 3 then begin  A := -1; B := -1;  end;
  if X = 4 then begin  A := -1; B := 0;   end;
  if (R + 5*A > 24)     or (R + 5*A < 1) or
     (C + 9*B + 9 > 80) or (C + 9*B < 1) then
    begin
       ClrScr;  Writeln ('OFF THE SCREEN');
    end
  else
    begin
      GotoXY (C + 8*B, R + 1*A);  Writeln ('G T E D S');
      GotoXY (C + 8*B, R + 2*A);  Writeln ('T       D');
      GotoXY (C + 8*B, R + 3*A);  Writeln ('E   ', X, '   E');
      GotoXY (C + 8*B, R + 4*A);  Writeln ('D       T');
      GotoXY (C + 8*B, R + 5*A);  Writeln ('S D E T G');
    end;
end.
```

```
{3.2}
program Thr2T93;
{ -- This program will solve an equation with +,-,*, or /. }
  var
    V1, V2, V3, S1, S2, X:  String[3];
    N1, N2, N3, I, J, Code: Integer;

begin
  Write ('Enter value: ');  Readln (V1);
  Write ('Enter symbol: '); Readln (S1);
  Write ('Enter value: ');  Readln (V2);
  Write ('Enter symbol: '); Readln (S2);
  Write ('Enter value: ');  Readln (V3);
  if S1 = '=' then begin
    S1 := S2;  S2 := '=';
    X := V1;   V1 := V2;  V2 := V3;  V3 := X;
  end;
  { -- Equation is now of the form: V1 [op] V2 = V3 }
  Val(V1, N1, Code);
  Val(V2, N2, Code);
  Val(V3, N3, Code);
  Write ('X = ');
  if S1 = '+' then
    if V1 = 'X' then
      Writeln (N3 - N2)
    else if V2 = 'X' then
      Writeln (N3 - N1)
    else
      Writeln (N1 + N2);

  if S1 = '-' then
    if V1 = 'X' then
      Writeln (N3 + N2)
    else if V2 = 'X' then
      Writeln (N1 - N3)
    else
      Writeln (N1 - N2);

  if S1 = '*' then
    if V1 = 'X' then
      Writeln (N3 div N2)
    else if V2 = 'X' then
      Writeln (N3 div N1)
    else
      Writeln (N1 * N2);

  if S1 = '/' then
    if V1 = 'X' then
      Writeln (N3 * N2)
    else if V2 = 'X' then
      Writeln (N1 div N3)
    else
      Writeln (N1 div N2);
end.
```

```
{3.3}
program Thr3T93;
{ -- This program prints combinations of digits summing to #. }
  var
    Digits:     String[7];
    Digit, A:   Array[1..7] of Byte;
    OneWritten: Boolean;
    I, J, Sum, NewSum, Code, Last, Total, Power: Integer;

begin
  Write ('Enter digits: ');  Readln (Digits);
  Write ('Enter sum: ');  Readln (Sum);
  NewSum := (Sum div 10) * 8 + (Sum mod 10);
  Last := Length(Digits);
  for I := 1 to Last do
    Val(Copy(Digits, I, 1), Digit[I], Code);
  for I := 1 to Last do
    A[I] := 0;

  Power := 1;
  for I := 1 to Last do Power := Power * 2;
  Power := Power - 1;

  for I := 1 to Power do begin
    J := 1;
    while (A[J] = 1) do begin
      A[J] := 0;
      Inc(J);
    end;
    A[J] := 1;
    Total := 0;
    for J := 1 to Last do
      if A[J] = 1 then
        Total := Total + Digit[J];
    if Total = NewSum then begin
      OneWritten := False;
      for J := 1 to Last do
        if A[J] = 1 then
          if OneWritten then
            Write ('+', Digit[J])
          else begin
            Write (Digit[J]);
            OneWritten := True;
          end;
      Writeln (' = ', Sum);
    end;  { -- if }
  end;  { -- for I }
end.
```

```
{3.4}
program Thr4T93;
{ -- This program will decompose a large integer into primes. }
  var
    A, Q:    Array[1..80] of Integer;
    LongNum: String[80];
    Prime, I, J, L, Num, Power, Code:  Integer;
    IsPrime, FirstFactor, QuotientIs0: Boolean;

procedure DisplayFactor;
{ -- This procedure will display a Factor raised to a power. }
  begin
    if FirstFactor then
      FirstFactor := False
    else
      Write(' * ');
    Write (Prime);
    if Power > 1 then
      Write ('^', Power);
    Power := 0;
  end;

procedure GetNextPrime;
{ -- This procedure will get the next prime to divide LongNum. }
  begin
    if Prime = 2 then
      Prime := 3
    else
      repeat
        Prime := Prime + 2;
        IsPrime := True;
        for J := 3 to Trunc(Sqrt(Prime)) do
          if Prime mod J = 0 then IsPrime := False;
      until IsPrime;
  end;

begin
  Write ('Enter number: ');  Readln (LongNum);
  L := Length(LongNum);
  for I := 1 to L do
    Val(Copy(LongNum, I, 1), A[I], Code);
  Prime := 2;  Power := 0;
  FirstFactor := True;  QuotientIs0 := False;
  repeat
    { -- Check if LongNum (Array A) is divisible by Prime. }
    Num := 0;
    for I := 1 to L do begin
      Num := Num * 10 + A[I];
      Q[I] := Num div Prime;
      Num := Num - Q[I] * Prime;
    end;
    if Num = 0 then  { -- Prime divided LongNum. }
      begin
        I := 1;
        while (Q[I] = 0) and (I <= L) do
```

```
          Inc(I);
        QuotientIs0 := (I = L) and (Q[L] = 1);
        L := L - I + 1;
        { -- Copy Quotient into array A to be divided again. }
        for J := 1 to L do
          A[J] := Q[J + I - 1];
        Inc(Power);
      end
    else    { -- Prime did not divide LongNum. }
      begin
        if Power >= 1 then
          DisplayFactor;
        GetNextPrime;
      end;  { -- else }
  until QuotientIs0;
  DisplayFactor;
end.



{3.5}
program Thr5T93;
{ -- This program will find words in 12 x 11 array of letters. }
  const
    LetRow: Array [1..12] of String[11] =
      ('DATAADFBAAM', 'JARBJCEDFOI', 'REAEEXEVDBC',
       'JESUSDEERNR', 'FABUUNMIEMO', 'LLMNSOIPTKC',
       'POQRSITRUOH', 'ABUVKWSXPPI', 'SOYZCPULMLP',
       'CCISABCDOAM', 'AEFGRHIJCRM', 'LKLETTEKSID');
   var
     I, J, L, Row, Col, FCol, LCol, FRow, LRow: Byte;
     LetCol: Array[1..11] of String[12];
     Word:   Array[1..2]  of String[12];

procedure DisplayCoordinates (FRow, FCol, LRow, LCol: Integer);
{ -- This procedure will display first and last letter coord. }
  begin
    Writeln ('FIRST LETTER: (', FRow: 2, ', ', FCol: 2, ')');
    Writeln ('LAST LETTER:  (', LRow: 2, ', ', LCol: 2, ')');
  end;

begin
  { -- String together the columns instead of Rows. }
  for I := 1 to 11 do begin
    LetCol[I] := '';
    for J := 1 to 12 do
      LetCol[I] := LetCol[I] + Copy(LetRow[J], I, 1);
  end;

  Write ('Enter word: ');  Readln (Word[1]);
  L := Length(Word[1]);

  { -- Reverse Word. }
  Word[2] := '';
  for I := 1 to L do
```

```
    Word[2] := Word[2] + Copy(Word[1], L - I + 1, 1);

  { -- Find words horizontally, (frontwards and backwards). }
  J := 0;
  repeat
    Inc(J);
    Row := 0;
    repeat
      Inc(Row);
      Col := Pos (Word[J], LetRow[Row]);
    until (Row = 12) or (Col > 0);
  until (Col > 0) or (J = 2);
  if Col > 0 then begin
    if J = 1 then begin
      FCol := 0;  LCol := L - 1;  end
    else begin
      FCol := L - 1;  LCol := 0;
    end;
    DisplayCoordinates (Row, Col + FCol, Row, Col + LCol);
    Exit;
  end;

  { -- Find words vertically, (frontwards and backwards). }
  J := 0;
  repeat
    Inc(J);
    Col := 0;
    repeat
      Inc(Col);
      Row := Pos (Word[J], LetCol[Col]);
    until (Col = 11) or (Row > 0);
  until (Row > 0) or (J = 2);
  if Row > 0 then begin
    if J = 1 then begin
      FRow := 0;  LRow := L - 1;  end
    else begin
      FRow := L - 1;  LRow := 0;
    end;
    DisplayCoordinates (Row + FRow, Col, Row + LRow, Col);
    Exit;
  end;
end.
```

```
{3.6}
program Thr6T93;
{ -- This program will solve two inequality equations. }
  var
    Eq1, Eq2, Op: String[3];
    S1, S2:       String[1];
    N1, N2, Code, Min, Max: Integer;

procedure Display (X, Y: Integer);
{ -- This procedure will display all integers between X and Y. }
  var
    I: Integer;

  begin
    Write (X);
    for I := X + 1 to Y do Write (',', I);
  end;

begin
  Write ('Enter equation 1: ');  Readln (Eq1);
  Write ('Enter logical op: ');  Readln (Op);
  Write ('Enter equation 2: ');  Readln (Eq2);
  S1 := Copy(Eq1, 2, 1);
  S2 := Copy(Eq2, 2, 1);
  Val (Copy(Eq1, 3, 1), N1, Code);
  Val (Copy(Eq2, 3, 1), N2, Code);

  if (S1 = '<') and (S2 = '>') and (Op = 'AND') and (N1 <= N2)
  or (S1 = '>') and (S2 = '<') and (Op = 'AND') and (N1 >= N2)
  then
    Writeln ('NO SOLUTION');

  if (S1 = '<') and (S2 = '>') and (Op = 'OR') and (N1 > N2)
  or (S1 = '>') and (S2 = '<') and (Op = 'OR') and (N1 < N2)
  then
    Writeln ('ALL INTEGERS');

  if N1 < N2 then
    begin
      Min := N1;  Max := N2;
    end
  else
    begin
      Min := N2;  Max := N1;
    end;

  { -- Check for finite solution, and if less than 6 integers. }
  if (S1 = '<') and (S2 = '>') and (Op = 'AND') and (N1 > N2)
  or (S1 = '>') and (S2 = '<') and (Op = 'AND') and (N1 < N2)
  then
    if Max - Min <= 7 then
      Display (Min + 1, Max - 1)
    else begin
      Display (Min + 1, Min + 3);
      Write ('...');
```

```
      Display (Max - 3, Max - 1);
    end;

  { -- Check for infinite # of negative solutions. }
  if (S1 = '<') and (S2 = '<') and (Op = 'AND') then begin
    Write ('...');
    Display (Min - 3, Min - 1);
  end;

  { -- Check for infinite # of positive solutions. }
  if (S1 = '>') and (S2 = '>') and (Op = 'AND') then begin
    Display (Max + 1, Max + 3);
    Write ('...');
  end;

  { -- Check for infinite # of positive and negitive solutions. }
  if (S1 = '>') and (S2 = '<') and (Op = 'OR') and (N1 > N2)
  or (S1 = '<') and (S2 = '>') and (Op = 'OR') and (N1 < N2) then
  begin
    Write ('...');
    Display (Min - 3, Min - 1);
    Write ('   ');
    Display (Max + 1, Max + 3);
    Write ('...');
  end;
end.


{3.7}
program Thr7T93;
{ -- This program will print the sum and product of 2 matrices. }
  const
    Base16: String[16] = '0123456789ABCDEF';
  var
    Mat:           Array[1..2, 1..3, 1..3] of LongInt;
    Sum, Prod:     LongInt;
    I, J, K, L, X: Integer;
    Num:           String[4];

procedure ConvertToBase16(N: LongInt);
{ -- This procedure will convert a Sum/Product element to B16. }
  var
    I, D:       Integer;
    Power:      LongInt;
    FirstDigit: Boolean;

begin
  Write (' ');
  FirstDigit := False;
  Power := 1;
  for I := 1 to 4 do Power := Power * 16;
  for I := 1 to 5 do begin
    D := Trunc(N / Power);
    if (D = 0) and not FirstDigit then
      Write(' ')
```

```
    else begin
      Write (Copy(Base16, D + 1, 1));
      FirstDigit := True;
    end;
    N := N - D * Power;
    Power := Power div 16;
  end;
end;


begin
  for I := 1 to 2 do begin
    for J := 1 to 3 do
      for K := 1 to 3 do begin
        Write ('Enter Mat', I, ' (', J, ',', K, '): ');
        Readln (Num);
        L := Length(Num);
        if L = 2 then
          Mat[I,J,K] := (Pos(Copy(Num,1,1), Base16) - 1) * 16
        else
          Mat[I,J,K] := 0;
        X := Pos(Copy(Num,L,1), Base16) - 1;
        Mat[I,J,K] := Mat[I,J,K] + X;
      end;
    Writeln;
  end;

  { -- Compute Sum }
  Write ('SUM =');
  for I := 1 to 3 do begin
    for J := 1 to 3 do begin
      Sum := Mat[1, I, J] + Mat[2, I, J];
      ConvertToBase16(Sum);
    end;
    Writeln;
    If I < 3 then Write (' ': 5);
  end;
  Writeln;

  { -- Compute Product }
  Write ('PRODUCT =');
  for I := 1 to 3 do begin
    for J := 1 to 3 do begin
      Prod := 0;
      for K := 1 to 3 do
        Prod := Prod + Mat[1, I, K] * Mat[2, K, J];
      ConvertToBase16(Prod);
    end;
    Writeln;
    if I < 3 then Write (' ': 9);
  end;
end.
```

```
{3.8}
program Thr8T93;
{ -- This program will find three 3-digit primes. }
  var
    P:          Array [1..200] of Integer;
    A:          Array [1..9]   of Integer;
    P1, P2, P3: String[3];
    PCat:       String[9];
    I, J, K, L, Num, Pnum, Sq, Sum,
    X, Tot, D1, D2, D3, D4, N2, Code: Integer;

begin
  Num := 101;  Pnum := 0;
  repeat
    Sq := Trunc(Sqrt(Num));
    I := 1;
    repeat
      I := I + 2;
    until (I > Sq) or (Num mod I = 0);
    if (I > Sq) then begin
      N2 := Num;
      D1 := N2 div 100;
      N2 := N2 - D1 * 100;
      D2 := N2 div 10;
      D3 := N2 - D2 * 10;
      if not ((D1 = 0) or (D2 = 0) or (D3 = 0)
      or (D1 = D2) or (D2 = D3) or (D1 = D3)) then begin
        Pnum := Pnum + 1;
        P[Pnum] := Num;
      end;
    end;

    Num := Num + 2;
  until (Num > 999);

  for I := 1 to Pnum - 2 do
    for J := I + 1 to Pnum - 1 do
      for K := J + 1 to Pnum do begin
        Tot := P[I] + P[J] + P[K];
        if Tot > 1234 then begin
          Str (P[I], P1);
          Str (P[J], P2);
          Str (P[K], P3);
          PCat := P1 + P2 + P3;
          for L := 1 to 9 do A[L] := 0;
          L := 0;
          repeat
            L := L + 1;
            Val(Copy(PCat, L, 1), X, Code);
            A[X] := A[X] + 1;
          until (L = 9) or (A[X] = 2);
          if A[X] < 2 then begin
            Sum := Tot;
            D1 := Sum div 1000;
            Sum := Sum - D1 * 1000;
```

```
            D2 := Sum div 100;
            Sum := Sum - D2 * 100;
            D3 := Sum div 10;
            D4 := Sum - D3 * 10;
            if (D1 < D2) and (D2 < D3) and (D3 < D4) then begin
              Write (P[I], ' + ', P[J], ' + ', P[K], ' = ');
              Writeln (Tot);
            end;
          end;  { -- for K }
        end;  { -- for J }
      end;  { -- for I }

end.



{3.9}
program Thr9T93;
{ -- This program will produce a binary search tree. }
uses Crt;
  const
    ColInc: Array[0..8] of Byte =
     (0, 15, 7, 3, 1, 0, 0, 0, 0);
  var
    Words: String[50];
    A:     Array[0..8, 1..256] of String[1];
    Ch:    String[1];
    I, J, R, C, Col, PrevCol: Integer;


begin
  Write ('Enter word(s): ');  Readln (Words);
  { -- Initialize tree to nulls. }
  for I := 0 to 8 do
    for J := 1 to 256 do
      A[I, J] := '';
  ClrScr;

  for I := 1 to Length(Words) do begin
    Ch := Copy (Words, I, 1);
    if Ch <> ' ' then begin
      R := 0;  C := 1;  Col := 40;
      { -- Traverse tree until an empty node exists. }
      while A[R, C] <> '' do begin
        if Ch <= A[R, C] then
          begin
            C := 2 * C - 1;
            Col := Col - ColInc[R + 1] - 1;
          end
        else
          begin
            C := 2 * C;
            PrevCol := Col;
            Col := Col + ColInc[R + 1] + 1;
```

```
          end;
        R := R + 1;
      end;  { -- While }
      A[R, C] := Ch;

      GotoXY(Col, R + 1);
      if R = 0 then  { -- Place first letter in center. }
        Write (Ch)
      else
        if C mod 2 = 1 then  { -- Place letter right of parent. }
          begin
            Write (Ch);
            for J := 1 to ColInc[R] do Write ('-');
            Write ('+');
          end
        else
          begin  { -- Place letter left of parent. }
            GotoXY(PrevCol, R + 1);
            Write ('+');
            for J := 1 to ColInc[R] do Write ('-');
            Write (Ch);
          end;
    end;  { -- if Ch }
  end;  { -- for I }
end.



{3.10}
program Thr10T93;
{ -- This program will determine the values F(X) converges. }
  var
    K, Inc, Factor,
    FX, FX0, FX1, FX2: Real;
    F:                 Array[1..5000] of Real;
    I, X, Iter:        Integer;
    Diverge, Found:    Boolean;

begin
  K := 0;
  for I := 1 to 2 do begin
    if I = 1 then Inc := 0.01 else Inc := 0.1;
    Diverge := False;  Factor := 1;  Found := False;
    while (K < 10) and not Found do begin
      K := K + Inc / Factor;
      X := 1;  F[X] := K;
      if Factor < 20 then
        Iter := 250 * Trunc(Factor)
      else
        Iter := 5000;
      while (X < Iter) and not Diverge do begin
        X := X + 1;
        F[X] := Exp(Ln(K) * F[X - 1]);
        Diverge := (F[X] > 9.9);
      end;
```

```
      if I = 1 then
        begin
          FX2 := FX1;  FX1 := FX0;  FX0 := F[X];
          if (FX2 > FX1) and (FX1 < FX0) then begin
            K := K - 2 * Inc / Factor;
            if (FX2 - FX1) < 0.0005 then begin
              Found := True;  FX := FX1;
            end;
            FX0 := FX2;  FX1 := FX0;
            Factor := Factor * 2;
          end;
        end
      else  { -- I = 2 }
        if Diverge then
          begin
            Diverge := False;
            K := K - Inc / Factor;
            if Inc/ Factor < 0.000005 then Found := True;
            Factor := Factor * 2;
          end
        else
          FX := F[X];
    end;  { -- While }

    if I = 1 then Write ('MINIMUM') else Write ('MAXIMUM');
    Write (' VALUE: ');
    if I = 1 then
      begin
        Write   ('F(X) = ', FX : 4:3, ' OCCURS WHEN ');
        Writeln ('K = ', K + Inc / Factor :4:3);
      end
    else
      begin
        Write   ('F(X) = ', FX : 2:1, '   OCCURS WHEN ');
        Writeln ('K = ', K + Inc / Factor :6:5);
      end;
  end;  { -- for I }
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '94 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T94;
{ -- This program will display the 1994 FHSCC sponsors. }
  var
    I: Integer;

begin
  Writeln ('FHSCC ''94 IS SPONSORED BY:');
  Writeln;
  for I := 1 to 4 do
    Writeln ('GTEDS  GTEDS  GTEDS  GTEDS  GTEDS');
  Writeln;
  for I := 1 to 4 do
    Writeln ('USF CENTER FOR EXCELLENCE');
  Writeln;
  for I := 1 to 4 do
    Writeln ('FLORIDA DEPARTMENT OF EDUCATION');
end.



{1.2}
program One2T94;
{ -- This program will determine if an applicant is hired. }
  var
    Entrance, Offer: String[8];

begin
  Write ('Entrance requirement: ');  Readln (Entrance);
  Write ('Plans to accept or reject offer: ');  Readln (Offer);
  Write ('APPLICANT WILL ');
  if (Entrance <> 'PASSED') or (Offer <> 'ACCEPT') then
    Write ('NOT ');
  Writeln ('BE HIRED');
end.


{1.3}
program One3T94;
{ -- This program will display number of employees. }
  var
    Current, Hiring, Leaving, Total: LongInt;

begin
  Write ('Enter current number: ');  Readln (Current);
  Write ('Enter number hiring: ');   Readln (Hiring);
  Write ('Enter number leaving: ');  Readln (Leaving);
  Total := Current + Hiring - Leaving;
  Writeln (Total, ' EMPLOYEES');
end.
```

```
{1.4}
program One4T94;
{ -- This program will total the millions converted. }
  var
    Num, Sum: Real;
    Million:  String[10];

begin
  Sum := 0;
  Write ('Enter number of accounts: ');  Readln (Num, Million);
  While Num > -999 do begin
    Sum := Sum + Num;
    Write ('Enter number of accounts: '); Readln (Num, Million);
  end;
  Sum := Sum + 0.00001;  { -- error factor of computer }
  if Sum - int(Sum) < 0.001 then
    Write (Trunc(Sum))
  else
    Write (Sum: 3:1);
  Writeln (' MILLION ACCOUNTS CONVERTED TO CBSS');
end.


{1.5}
program One5T94;
{ -- This program will compute the gross wages earned. }
  var
    Hours, Rate, OverTime, Wages: Real;

begin
  Write ('Enter hours, rate: ');  Readln (Hours, Rate);
  if Hours > 40 then
    Hours := Hours + (Hours - 40) * 0.5;
  Wages := Hours * Rate;
  Writeln ('GROSS WAGES ARE $', Wages :5:2);
end.
```

```
{1.6}
program One6T94;
{ -- This program will tally the number of customers sold. }
  var
    AreaCode, Num, I: Integer;
    Sum:              LongInt;

begin
  Write ('Enter number of area codes: ');  Readln (Num);
  Sum := 0;
  for I := 1 to Num do begin
    Write ('Enter area code: ');  Readln (AreaCode);
    Case AreaCode of
      706:  Inc(Sum, 95000);
      208:  Inc(Sum, 54321);
      912:  Inc(Sum, 99825);
      605:  Inc(Sum, 88776);
      404:  Inc(Sum, 90175);
    end;
  end;
  Writeln ('TOTAL NUMBER OF ACCOUNTS BEING SOLD = ', Sum);
end.

{1.7}
program One7T94;
{ -- This program will display the cost to fix error in phase. }
  const
    Phases: Array[1..6] of String[15] =  ('REQUIREMENTS',
       'DESIGN', 'CODING', 'SYSTEM TEST', 'ACCEPTANCE TEST',
       'MAINTENANCE');
    Factor: Array [1..6] of Integer = (1, 5, 10, 20, 50, 100);
  var
    I, Cost: Integer;
    Phase:   String[15];

begin
  Write ('Enter cost $: ');  Readln (Cost);
  Write ('Enter phase: ');   Readln (Phase);
  I := 1;
  while Phase <> Phases[I] do Inc(I);
  Write ('COST IS $', Cost * Factor[I]);
  Writeln (' TO FIX PROBLEM IN ', Phase, ' PHASE');
end.

{1.8}
program One8T94;
{ -- This program will compute the maximum blocksize. }
  var
    LRecL, Num: Integer;
begin
  Write ('Enter logical record length: ');  Readln (LRecL);
  Num := 23476 div LRecL;
  Writeln ('BLOCKSIZE = ', LRecL * Num, ' BYTES');
end.
```

```
{1.9}
program One9T94;
{ -- This program will compute an electric bill. }
  var
    Hours, Bill, Rate: Real;

begin
  Write ('Enter kilowatt hours: ');  Readln (Hours);
  if Hours < 10.0 then
    Rate := 4.95
  else
    Rate := 5.65;
  Bill := Rate * Hours;
  Bill := Bill * (1 + 0.03 + 0.06);
  if Hours > 30.0 then
    Bill := Bill + 25.0;
  Writeln ('THE CUSTOMER''S BILL IS $', Bill :3:2);
end.


{1.10}
program One10T94;
{ -- This program will determin if a 5x5 matrix is symmetric. }
  var
    A:         Array[1..5, 1..5] of Integer;
    I, J:      Integer;
    Symmetric: Boolean;

begin
  for I := 1 to 5 do begin
    Write ('Enter row: ');
    Readln (A[I,1], A[I,2], A[I,3], A[I,4], A[I,5]);
  end;
  Symmetric := True;
  for I := 1 to 5 do
    for J := 1 to 5 do
      if A[I, J] <> A[J, I] then
        Symmetric := False;

  Write ('MATRIX IS ');
  if not Symmetric then
    Write ('NOT ');
  Writeln ('SYMMETRIC');
end.
```

```
{2.1}
program Two1T94;
{ -- This program will simulate NTF's ESP utility. }
  var
    Jobs:          String[50];
    Job:           Array[1..20] of String[2];
    I, L, LastCK: Integer;
    OK:            String[1];

begin
  Write ('Enter jobs/CK: ');  Readln (Jobs);
  L := (Length(Jobs) + 1) div 3;
  for I := 1 to L do
    Job[I] := Copy(Jobs, I*3 - 2, 2);

  LastCK := 0;
  repeat
    I := LastCK + 1;
    while Job[I] <> 'CK' do begin
      Writeln (Job[I]);
      Inc(I);
    end;
    Writeln ('EVERYTHING OK?' );  Readln (OK);
    if OK = 'N' then
      I := LastCK
    else
      LastCK := I;
  until I = L;
end.
```

```
{2.2}
program Two2T94;
{ -- This program will display random letters in random areas. }
  uses Crt;
  var
    Letter, LastLet, Ch: Char;
    R, C:                Integer;

begin
  Randomize;
  Ch := ' ';
  repeat
    ClrScr;
    if Ch = ' ' then begin
        Letter := Chr(65 + Random(26));
        Ch := Letter;
      end
    else
      Letter := Ch;
    LastLet := Letter;
    repeat
      R := Random(23) + 1;  C := Random(79) + 1;
      GotoXY (C, R);  Write (Letter);
      Delay(100);
      if Keypressed then
        Ch := UpCase(ReadKey);
    until (Ch <> LastLet);
  until (Ch <> ' ') and ((Ch < 'A') or (Ch > 'Z'));
end.
```

```
{2.3}
program Two3T94;
{ -- This program will transliterate Hebrew to English. }
  var
    St, Trans:  String[80];
    I:          Integer;
    Ch, LastCh: String[1];
    Let:        String[2];

begin
  Write ('Enter letters: ');  Readln (St);
  LastCh := ' ';  Trans := '';
  for I := 1 to Length(St) do begin
    Ch := Copy(St, I, 1);  Let := Ch;
    if LastCh = ' ' then begin
      if Ch = 'A' then
        if Copy(St, I+1, 1) = 'L' then
          Let := ')'
        else
          Let := '(';
      if Copy(St, I, 3) = 'HET' then Let := 'CH';
      if Copy(St, I, 2) = 'TS'  then Let := 'TS';
      Trans := Let + Trans;
    end;
    LastCh := Ch;
  end;
  Writeln (Trans);
end.
```

```
{2.4}
program Two4T94;
{ -- This program will append a "security digit" to an account }
  var
    Acct:  String[15];
    Ch:    String[1];
    Error: Boolean;
    Sum, I, L, Dig, Code: Integer;

begin
  Write ('Enter account number: ');  Readln (Acct);
  L := Length (Acct);
  Error := False;
  if (L <> 7) and (L <> 9) then begin
    Writeln ('ERROR - INCORRECT LENGTH');
    Error := True;
  end;

  { -- Sum the valid digits }
  Sum := 0;
  for I := 1 to L do begin
    Ch := Copy(Acct, I, 1);
    Val (Ch, Dig, Code);
    if (Dig = 0) and (Ch <> '0') then begin
      Writeln ('ERROR - NON-NUMERIC');
      Exit;
    end;
    Sum := Sum + Dig;
  end;

  { -- If account is valid, append security digit }
  if not Error then begin
    Write (Acct);
    if Sum mod 2 = 0 then
      Writeln ('1')
    else
      Writeln ('0');
  end;
end.
```

```
{2.5}
program Two5T94;
{ -- This program will count the digits used in a book. }
  var
    I, J, LPage, M, Dig, Code, Max, Min: Integer;
    A:    Array[0..9] of Integer;
    Page: String[4];

begin
  Write ('Enter last page: ');  Readln (LPage);
  Write ('Enter M: ');          Readln (M);
  for I := 0 to 9 do A[I] := 0;

  for I := 2 to LPage do begin
    if (I mod M > 0) then begin
      Str (I, Page);
      for J := 1 to Length(Page) do begin
        Val (Copy(Page, J, 1), Dig, Code);
        Inc(A[Dig]);
      end;
    end;
  end;
  Max := 0;  Min := 32000;
  for I := 0 to 9 do begin
    Writeln (I, ' APPEARS ', A[I], ' TIMES');
    if A[I] > Max then Max := A[I];
    if A[I] < Min then Min := A[I];
  end;

  Writeln;
  Write ('DIGIT(S) APPEARING THE MOST: ');
  for I := 0 to 9 do
    if A[I] = Max then Write (I, ' ');
  Writeln;
  Write ('DIGIT(S) APPEARING THE LEAST: ');
  for I := 0 to 9 do
    if A[I] = Min then Write (I, ' ');
end.
```

```
{2.6}
program Two6T94;
{ -- This program will compute the roots for a quadratic. }
  var
    A, B, C, D, R1, R2: Integer;

begin
  Write ('Enter coefficients A, B, C: ');  Readln (A, B, C);
  D  := B * B - 4 * A * C;
  Write ('THE ROOTS ARE ');
  if D >= 0 then
    begin
      Writeln ('REAL');
      R1 := (-B + Trunc(Sqrt(D))) div (2 * A);
      R2 := (-B - Trunc(Sqrt(D))) div (2 * A);
      if D > 0 then
        Writeln ('THE ROOTS ARE ', R1, ' AND ', R2)
      else
        Writeln ('THE ONLY ROOT IS ', R1);
    end
  else  { -- D < 0    Roots are Complex }
    begin
      Writeln ('COMPLEX');
      R1 := -B div (2 * A);
      R2 := Trunc(Sqrt(-D)) div (2 * A);
      Write ('THE ROOTS ARE ', R1, ' + ', R2, 'I AND ');
      Writeln (R1, ' - ', R2, 'I');
    end;
end.
```

```
{2.7}
program Two7T94;
{ -- This program will generate 5 customer account numbers. }
  const
    Num: Integer = 15;
  var
    Seed: Real;
    I, J, Dig, Code, Sum, CheckDig: Integer;
    Cust: String[10];
    Temp: String[1];

begin
  Write ('Enter seed used last: ');  Readln (Seed);
  I := 0;
  while I < Num do begin
    { -- Add 1 and reverse last 2 digits }
    Seed := Seed + 1;
    Str (Seed :9:0, Cust);

    Temp := Cust[9];  Cust := Copy(Cust, 1, 8);
    Insert (Temp, Cust, 8);

    for J := 1 to 9 do
      if Cust[J] = ' ' then Cust[J] := '0';
    { -- Shift digits 3-9 and insert last 2 digits }
    Cust := Copy(Cust, 1, 2) + Copy(Cust, 8, 2) +
            Copy(Cust, 3, 5);
    { -- Calculate Check Digit }
    Sum := 0;
    for J := 1 to 9 do begin
      Val (Copy(Cust, J, 1), Dig, Code);
      Sum := Sum + Dig * (11 - J);
    end;
    CheckDig := 11 - (Sum mod 11);
    if CheckDig = 11 then CheckDig := 0;
    if CheckDig <> 10 then begin
      Writeln (Cust, CheckDig);
      Inc(I);
    end;
  end;  { -- while }
end.
```

```
{2.8}
program Two8T94;
{ -- This program will compute speed, distance, time. }
  var
    S, D, T, HH, MM: Real;
    Tim:             String[6];
    Ttype:           String[1];
    L, Code:         Integer;

begin
  Write ('Enter speed, distance: ');  Readln (S, D);
  Write ('Enter time: ');  Readln (Tim);
  if Tim <> '0' then begin
    L := Length(Tim);
    Ttype := Copy(Tim, L, 1);
    if (TType = 'H') or (TType = 'M') then
      Val(Copy(Tim, 1, L-1), T, Code)
    else { -- Ttype = 'C' }
      begin
        Val (Copy(Tim,1,2), HH, Code);
        Val (Copy(Tim,4,2), MM, Code);
        T := HH + MM / 60;
      end;
    if Ttype = 'M' then
      T := T / 60;
  end;

  if S = 0 then
      Writeln ('SPEED = ', D / T :5:1, ' MPH')
  else if D = 0 then
      Writeln ('DISTANCE = ', S * T :6:1, ' MILES')
  else if Tim = '0' then
      Writeln ('TIME = ', D / S :4:2, ' HOURS');
end.
```

```
{2.9}
program Two9T94;
{ -- This program will compute the response time. }
  var
    RDate, CDate, RTime, CTime:           String[8];
    RDay, CDay, RMin, RHour, CMin, CHour: Byte;
    Code, Res:                            Integer;

begin
  Write ('Enter reported date: ');  Readln (RDate);
  Write ('Enter reported time: ');  Readln (RTime);
  Write ('Enter cleared date: ');   Readln (CDate);
  Write ('Enter cleared time: ');   Readln (CTime);

  Val(Copy(RDate, 4, 2), RDay, Code);
  Val(Copy(CDate, 4, 2), CDay, Code);
  Val(Copy(RTime, 1, 2), RHour, Code);
  Val(Copy(RTime, 4, 2), RMin, Code);
  Val(Copy(CTime, 1, 2), CHour, Code);
  Val(Copy(CTime, 4, 2), CMin, Code);

  Res := 0;
  if RHour < 8 then begin
    RHour := 8;  RMin := 0;
  end;
  if CHour < 8 then begin
    CHour := 8;  CMin := 0;
  end;
  if CHour >= 17 then begin
    CHour := 17;  CMin := 0;
  end;
  if RHour >=17 then begin
    RHour := 17;  RMin := 0;
  end;

  Res := (CDay - RDay) * 9 * 60;
  Res := Res + (CHour - RHour) * 60 + (CMin - RMin);

  Writeln ('RESPONSE TIME WAS ', Res, ' MINUTES');
end.
```

```
{2.10}
program Two10T94;
{ -- This program will display the discounts for calling plans }
  var
    OrigNum, ToNum:                    String[10];
    Handicap, OrigArea, ToArea:        String[3];
    CallLen, Cost, PlanA, PlanB, PlanC: Real;

begin
  Write ('Enter originating number: ');  Readln (OrigNum);
  Write ('Enter number called: ');  Readln (ToNum);
  Write ('Handicapped person?: ');  Readln (Handicap);
  Write ('Enter length of call: '); Readln (CallLen);
  Write ('Enter cost of call $: '); Readln (Cost);

  PlanA := 9E9;  PlanB := 9E9;  PlanC := 9E9;
  OrigArea := Copy(OrigNum, 1, 3);
  ToArea   := Copy(ToNum, 1, 3);
  if (CallLen >= 5.0) and (OrigArea <> ToArea) then begin
    PlanA := Cost * 0.85;
    Writeln ('THE PLAN A CHARGE WOULD BE $', PlanA :3:2);
  end;
  if Handicap = 'YES' then begin
    PlanB := Cost * 0.90;
    Writeln ('THE PLAN B CHARGE WOULD BE $', PlanB :3:2);
  end;
  if (ToArea = '407') and (OrigArea <> ToArea)
  and (CallLen >= 3.5) then begin
    PlanC := Cost * 0.8775;
    Writeln ('THE PLAN C CHARGE WOULD BE $', PlanC :3:2);
  end;

  if (PlanA = 9E9) and (PlanB = 9E9) and (PlanC = 9E9) then
    Writeln ('THIS PERSON DOES NOT QUALIFY FOR ANY PLANS')
  else begin
    Write ('THIS PERSON WOULD RECEIVE PLAN ');
    if (PlanA < PlanB) and (PlanA < PlanC) then
      Writeln ('A')
    else
      if (PlanB < PlanA) and (PlanB < PlanC) then
         Writeln ('B')
       else
         Writeln ('C');
  end;
end.
```

```
{3.1}
program Thr1T94;
{ -- This program will convert transliterated English to Greek. }
{ -- The Greek letters ETA and OMICRON are not used. }
{ -- The Greek letter THETA is placed at the end of the list. }
  const
    Name: Array [1..24] of String[8] = ('ALPHA', 'BETA', 'GAMMA',
      'DELTA', 'EPSILON', 'ZETA', '-TA',  'IOTA',
      'KAPPA', 'LAMBDA', 'MU', 'NU', 'XI', '-MICRON', 'PI',
      'RHO', 'SIGMA', 'TAU', 'UPSILON', 'PHI', 'CHI', 'PSI',
      'OMEGA', 'THETA');
    Value: Array [1..24] of Integer = (1, 2, 3, 4, 5, 7, 8,
      10, 20, 30, 40, 50, 60, 70, 80,
      100, 200, 300, 400, 500, 600, 700, 800, 9);
  var
    I, J, Sum, Inc: Integer;
    Trans:          String[15];
    Ch:             String[2];

begin
  Write ('Enter transliteration: ');  Readln (Trans);
  Sum := 0;
  I := 1;
  while I <= Length(Trans) do begin
    Ch := Copy(Trans, I, 2);
    if (Ch = 'TH') or (Ch = 'PH') or (Ch = 'CH') or (Ch = 'PS')
    then
      Inc := 2
    else
      Inc := 1;
    J := 1;
    while Copy(Trans, I, Inc) <> Copy(Name[J], 1, Inc) do
      J := J + 1;
    Write (Name[J], ' ');
    Sum := Sum + Value[J];
    I := I + Inc;
  end;  { -- While I }
  Writeln;  Writeln ('NUMERICAL SUM = ', Sum);
end.


{3.2}
program Thr2T94;
{ -- This program will move a taxi in a grid. }
  const
    South: Integer = 8;
  var
    Num, SNum, NumLet, SNumLet: Integer;
    SLet, Dir:   Char;
    Out, TooFar: Boolean;

begin
  Write ('Enter starting position: ');  Readln (SLet, SNum);
  Num := SNum;
  SNumLet := Ord(SLet) - Ord('A') + 1;  NumLet := SNumLet;
```

```
repeat
  Write ('Enter direction: ');  Readln (Dir);
  Out := False;  TooFar := False;
  Case Dir of
    'N':  if Num = 1 then
            Out := True
          else
            if SNum - 2 = Num then
              TooFar := True
            else
              Dec(Num);
    'S':  if Num = South then
            Out := True
          else
            if SNum + 2 = Num then
              TooFar := True
            else
              Inc(Num);
     'W':  if NumLet = 1 then
             Out := True
           else
             if SNumLet - 2 = NumLet then
               TooFar := True
             else
               Dec(NumLet);
     'E':  if NumLet = 26 then
             Out := True
           else
             if SNumLet + 2 = NumLet then
               TooFar := True
             else
               Inc(NumLet);
    end;  { -- case }

    if Out then
      Writeln ('LOCATION IS OUTSIDE CITY LIMITS')
    else if TooFar then
      begin
        Write ('LOCATION IS TOO FAR ');
        Case Dir of
          'N':  Writeln ('NORTH');
          'S':  Writeln ('SOUTH');
          'W':  Writeln ('WEST');
          'E':  Writeln ('EAST');
        end;
      end
    else
      if Dir <> 'Q' then begin
        Write ('TAXI LOCATION IS ');
        Writeln (Chr(NumLet + 64), ',', Num);
      end;
   until Dir = 'Q';
 end.
```

```
{3.3}
program Thr3T94;
{ -- This program will display anagrams. }
  var
    W, W2: Array [1..9] of String[7];
    SortW: Array [1..7] of String[1];
    I, J, K, L, Num, Tot:  Integer;
    T:                     String[7];

begin
  Write ('Enter number of words: ');  Readln (Num);
  for I := 1 to Num do begin
    Write ('Enter word: ');  Readln (W[I]);
  end;

  { -- Sort words in ascending order }
  for I := 1 to Num - 1 do
    for J := I + 1 to Num do
      if W[I] > W[J] then begin
        T := W[I];  W[I] := W[J];  W[J] := T;
      end;

  { -- Sort letters within word and store in W2[] }
  for I := 1 to Num do begin
    L := Length(W[I]);
    for J := 1 to L do
      SortW[J] := Copy(W[I], J, 1);
    for J := 1 to L - 1 do
      for K := J + 1 to L do
        if SortW[J] > SortW[K] then begin
          T := SortW[J];  SortW[J] := SortW[K];
          SortW[K] := T;
        end;

    W2[I] := '';
    for J := 1 to L do
      W2[I] := W2[I] + SortW[J];
  end;

  { -- Compare every pair of sorted words for a match. }
  Tot := 0;
  for I := 1 to Num - 1 do
    for J := I + 1 to Num do
      if W2[I] = W2[J] then begin
        Tot := Tot + 1;
        if Tot = 1 then
          Write ('ANAGRAMS: ')
        else
          Write ('          ');
        Writeln (W[I], ', ', W[J])
      end;
  if Tot = 0 then
    Writeln ('NO ANAGRAMS IN LIST');
end.
```

```
{3.4}
program Thr4T94;
{ -- This program will place money in envelopes. }
  var
    Money, A, B, C, D, Incr, Total: Integer;

begin
  Write ('Enter amount of money: '); Readln (Money);
  Total := 0;
  Incr := Money div 2;
  for A := 1 to Incr - 2 do
    for B := A + 1 to Incr - 1 do
      for C := B + 1 to Incr do begin
        { -- D will contain the largest amount to disperse }
        D := Money - A - B - C;
        if (A < B) and (B < C) and (C < D) then begin
          Write ('TAKE ', A, ' ', B, ' ', C, ' ', D);
          { -- (D - A) dollars are dispersed to make }
          { --         A=B, B=C, C=D, and D=A        }
          Write (' AND DISPERSE ', D - A, ' DOLLARS TO MAKE ');
          Writeln (B, ' ', C, ' ', D, ' ', A);
          Inc(Total);
        end;
      end;
   Writeln ('TOTAL NUMBER OF SOLUTIONS = ', Total);
end.
```

```
{3.5}
program Thr5T94;
{ -- This program will convert Gregorian and Julian dates. }
  const
    Month: Array [1..12] of Byte =
      (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31);
  var
    MDays: Array [1..12] of Byte;
    M, D, Y, I, Days, Code: Integer;
    DType, Date, YY:        String[11];

begin
  for I := 1 to 12 do MDays[I] := Month[I];
  Write ('Enter Julian or Gregorian: ');  Readln (DType);
  Write ('Enter date: ');  Readln (Date);
  if DType = 'GREGORIAN' then
    { -- Convert Gregorian to Julian }
    begin
      Val(Copy(Date, 1, 2), M, Code);
      Val(Copy(Date, 4, 2), D, Code);
      YY := Copy(Date, 7, 2);
      Val(YY, Y, Code);
      Days := D;
      for I := 1 to M - 1 do
        Days := Days + MDays[I];
      if (Y mod 4 = 0) and (M > 2) then
        Inc(Days);
      Write ('JULIAN DATE = ', YY);
      if Days < 100 then Write ('0');
      if Days < 10 then Write ('0');
      Writeln (Days)
    end
  else  { -- Convert Julian to Gregorian }
    begin
      YY := Copy(Date, 1, 2);
      Val(YY, Y, Code);
      Val(Copy(Date, 3, 3), D, Code);
      M := 1;
      if Y mod 4 = 0 then
        MDays[2] := 29;
      while D > MDays[M] do begin
        D := D - MDays[M];
        Inc(M);
      end;
      Write ('GREGORIAN DATE = ');
      if M < 10 then Write ('0');
      Write (M, '/');
      if D < 10 then Write ('0');
      Write (D, '/');
      Writeln (YY);
    end;
end.
```

```
{3.6}
program Thr6T94;
{ -- This program to convert a number for one base to another. }
  var
    Base1, Base2, Num1V, Num2, Power: Real;
    I, J, K, X, Digit:                Byte;
    Num1, NumOut:                     String[9];

begin
  Write ('Enter base of first number: ');  Readln (Base1);
  Write ('Enter number: ');                Readln (Num1);
  Write ('Enter base of output: ');        Readln (Base2);

  { -- Convert Num1 to base 10 number Num1V }
  Num1V := 0;
  for I := 1 to Length(Num1) do begin
    Digit := Ord(Num1[I]) - Ord('0');
    if Digit > 9 then    { -- Digit is a letter digit }
      Dec(Digit, 7);
    Power := 1;
    for J := 1 to Length(Num1) -I do
      Power := Power * Base1;
    Num1V := Num1V + Digit * Power;
  end;

  { -- Convert Num1V to Base2 number }
  NumOut := '';
  J := Trunc(Ln(Num1V) / Ln(Base2));
  for I := J downto 0 do begin
    Power := 1;
    for K := 1 to I do  Power := Power * Base2;
    X := Trunc(Num1V / Power);
      NumOut := Copy ('0123456789ABCDEF', X + 1, 1) + NumOut;
    Num1V := Num1V - X * Power;
  end;
  Writeln (NumOut);
end.
```

```
{3.7}
program Thr7T94;
{ -- This program will SHELL sort numbers generated. }
  const
    Num: Integer = 8000;
    Max: Integer = 7;
  var
    I, J, P, Last, Increment: Integer;
    X:                        Array [-1093..8000] of Real;
    Incr:                     Array [1..7] of Integer;
    Pow, Q, Temp, T:          Real;

begin
  Write ('Enter seed X[0]: ');  Readln (X[0]);
  Pow := 1;
  for I := 1 to 20 do Pow := Pow * 2;
  for I := 1 to Num do begin
    Q := Int ((69069.0 * X[I-1]) / Pow);
    X[I] := 69069.0 * X[I-1] - Pow * Q;
  end;
  for I := -1093 to -1 do X[I] := 0;

  { -- SHELL SORT ROUTINE }
  Incr[Max] := 1;
  { -- Compute Increments }
  for I := Max - 1 downto 1 do Incr[I] := 3 * Incr[I+1] + 1;
  for I := 1 to Max do begin
    Increment := Incr[I];
    for J := 1 to Increment do begin
      Last := Increment + J;
      while Last <= Num do begin
        P := Last;
        T := X[P];
        X[1 - Increment] := T;
        while T < X[P - Increment] do begin
          X[P] := X[P - Increment];
          Dec(P, Increment);
        end;
        X[P] := T;
        Inc(Last, Increment);
      end;
    end;  { -- for J }
  end;  { -- for I }

  { -- Display every 1000th number in ascending order }
  for I := 1 to Num div 1000 do
    Writeln (I*1000, 'TH NUMBER = ', X[I*1000]: 6:0);
end.
```

```
{Alternate solution to 3.7}
program Thr7T94;
{ -- This program will QUICK sort numbers generated. }
  const
    Num: Integer = 8000;
  var
    I:      Integer;
    X:      Array [0..8000] of Real;
    Pow, Q: Real;

procedure Quicksort(L, R: Integer);
{ -- sorts global array X[L..R] where X[R + 1] > any X[L..R] }
  var
    I, J:   Integer;
    T, Piv: Real;

begin
  if L < R then begin
    I := L + 1;  J := R;  Piv := X[L];
    repeat  { -- move pointers I, J inwards as far as possible }
      while X[I] <= Piv do I := I + 1;
      while X[J] > Piv  do J := J - 1;
      if I {still} < J then begin
        { -- Exchange items pointed to by I and J }
        T := X[I];  X[I] := X[J];  X[J] := T;
      end;
    until I > J;
    { -- Now two final replacements finish a partition }
    X[L] := X[J];  X[J] := Piv;
    { -- Finish by recursively sorting left, right partitions }
    Quicksort (L, J-1);  Quicksort (I, R);
  end;  { -- if }
end;  { procedure Quicksort }

begin
  Write ('Enter seed X[0]: ');  Readln (X[0]);
  Pow := 1;
  for I := 1 to 20 do Pow := Pow * 2;
  for I := 1 to Num do begin
    Q := Int ((69069.0 * X[I-1]) / Pow);
    X[I] := 69069.0 * X[I-1] - Pow * Q;
  end;

  Quicksort (1, Num);

  { -- Display every 1000th number in ascending order }
  for I := 1 to Num div 1000 do
    Writeln (I*1000, 'TH NUMBER = ', X[I*1000]: 6:0);
end.
```

```
{3.8}
program Thr8T94;
{ -- This program will compute the volume of a sphere using PI }
  const
    PI1: String[37] = '31415926535897932384626433832795028884';
    PI2: String[37] = '19716939937510582097494459230781640 62';
    PI3: String[37] = '86208998628034825342117067982148086 51';
  var
    Prod: Array[1..120] of Integer;
    A:    Array[1..4]   of Integer;
    PI:   String[111];
    C, CC, I, J, K, L, N, Pr, R, Radius, Code: Integer;

begin
  Write ('Enter N: ');  Readln (N);
  Write ('Enter radius: ');  Readln (Radius);

  { -- Assign digits of PI to Array PI[ ] }
  PI := PI1 + PI2 + PI3;
  L := Length(PI);
  for I := 1 to L do
    Val(Copy(PI, L - I + 1, 1), Prod[I], Code);

  for I := 1 to 3 do A[I] := Radius;
  A[4] := 4;
  C := 0;
  { -- Multiply PI by Radius (3 times) then by 4. }
  for I := 1 to 4 do begin
    for J := 1 to L do begin
      Prod[J] := Prod[J] * A[I] + C;
      C := Prod[J] div 10;
      Prod[J] := Prod[J] - C * 10;
    end;
    while C > 0 do begin
      CC := C div 10;
      Inc(L);
      Prod[L] := C - CC * 10;
      C := CC;
    end;
  end;
  { -- Divide the product by 3. }
  R := 0;
  for I := L downto 1 do begin
    Pr      := Prod[I] + R * 10;
    Prod[I] := Pr div 3;
    R       := Pr - Prod[I] * 3;
  end;
  if Prod[L] = 0 then Dec(L);
  { Display the Volume with the decimal point. }
  for I := L downto 111 - N do begin
    if I = 110 then Write ('.');
    Write (Prod[I]);
  end;
end.
```

```
{3.9}
program Thr9T94;
{ -- This program will display the barcode of an address. }
  const
    Val: Array[1..5] of Byte = (7, 4, 2, 1, 0);
  var
    I, J, L, P, NumBars, CheckDig, Sum, Dig: Byte;
    Addr1, Addr2: String[30];
    BarCode:      String[14];
    Zip4, DPoint: String[4];

begin
  Write ('Enter address 1: ');  Readln (Addr1);
  Write ('Enter address 2: '); Readln (Addr2);

  { -- Extract Zip+4 or Zip from 2nd line of address }
  L := Length (Addr2);
  I := L;
  while Copy(Addr2, I, 1) <> ' ' do I := I - 1;
  if L - I = 10 then
    BarCode := Copy(Addr2, I + 1, 5) + Copy (Addr2, L - 3, 4)
  else
    BarCode := Copy(Addr2, L - 4, 5);

  { -- Extract possible Zip+4 and/or next 2 Delivery points }
  Zip4 := '';
  if Copy(Addr1, 1, 8) = 'P.O. BOX' then
    begin
      L := Length (Addr1);
      I := L;
      while Copy(Addr1, I, 1) <> ' ' do I := I - 1;
      for J := 1 to 4 - (L - I) do
        Zip4 := Zip4 + '0';
      Zip4 := Zip4 + Copy(Addr1, I + 1, L - I);
      DPoint := Copy(Zip4, 3, 2);
    end
  else
    begin
      Zip4 := '0000';
      Addr1 := '0' + Addr1;
      P := Pos (' ', Addr1);
      DPoint := Copy (Addr1, P - 2, 2);
    end;

  if Length(BarCode) = 5 then
    BarCode := BarCode + Zip4;
  BarCode := BarCode + DPoint;

  { -- Calculate Check Digit for 12-digit Barcode and display }
  Sum := 0;
  for I := 1 to 11 do
    Sum := Sum + Ord(BarCode[I]) - 48;
  CheckDig := 10 - (Sum mod 10);
  if CheckDig = 10 then CheckDig := 0;
  Barcode := BarCode + Chr(CheckDig + 48);
```

```
  Writeln (' ': 12, 'DELIVERY POINT BAR CODE = ', BarCode);
  Writeln;

  { -- Display Frame bars and encoded BarCode }
  Write ('!');
  for I := 1 to 12 do begin
    Dig := Ord(BarCode[I]) - 48;
    NumBars := 0;
    if Dig = 0 then  { -- exception for 0 = 7 + 4}
      Dig := 11;
    for J := 1 to 5 do
      if (Dig >= Val[J]) and (NumBars < 2) then
        begin
          Write ('!');
          Dig := Dig - Val[J];
          NumBars := NumBars + 1;
        end
      else
        Write (' ');
  end;  { -- for I }
  Writeln ('!');

  for I := 1 to 62 do Write ('!');
end.
```

```
{3.10}
program Thr10T94;
{ -- This program produces a 3 x 3 magic square. }
  type
    String9  = Array [1..9] of Integer;
  var
    I, Number, FNum, Inc, MNum, Sum:  Integer;
    Num1, Num2, Row, Col, Pos1, Pos2: Integer;
    S: String9;

procedure Permute ({Using}     N:     Integer;
                   {Giving} var S:     String9);
{ -- This procedure will interchange the elements in Array S. }
  var
    I, J, K, Temp: Integer;
    MagicNum:      Boolean;

begin
  If N > 1 then
    begin
      Permute (N - 1, S);
      for I := N-1 downto 1 do begin
        {Interchange the elements in S[N] and S[I] }
        Temp := S[N];  S[N] := S[I];  S[I] := Temp;
        Permute (N - 1, S);
        Temp := S[N];  S[N] := S[I];  S[I] := Temp;
      end;  { -- for I }
    end  { -- if then }
  else
    if (S[Pos1] = Num1) and (S[Pos2] = Num2) then begin
      MagicNum := True;
      { -- Check if Row elements sum to Magic Number. }
      for J := 0 to 2 do
        if S[J*3 + 1] + S[J*3 + 2] + S[J*3 + 3] <> MNum then
          MagicNum := False;
      { -- Check if Column elements sum to Magic Number. }
      if MagicNum then
        for J := 1 to 3 do
          if S[J] + S[J + 3] + S[J + 6] <> MNum then
            MagicNum := False;
      { -- Check if Diagonal elements sum to Magic Number. }
      if MagicNum then
        if  (S[1] + S[5] + S[9] = MNum)
        and (S[3] + S[5] + S[7] = MNum) then begin
          { -- Display the Magic Square. }
          for J := 0 to 2 do begin
            for K := 1 to 3 do
              Write (S[J * 3 + K] :3);
            Writeln;
          end;
          Writeln;
        end;
    end;  { -- if S[Pos1] }
end;  { -- procedure}
```

```
{ -- Main program }
begin
  Write ('Enter first number: ');  Readln(FNum);
  Write ('Enter increment: ');     Readln(Inc);

  Write ('Enter number: ');    Readln (Num1);
  Write ('Enter row, col: ');  Readln (Row, Col);
  Pos1 := (Row - 1) * 3 + Col;
  Write ('Enter number: ');    Readln (Num2);
  Write ('Enter row, col: ');  Readln (Row, Col);
  Pos2 := (Row - 1) * 3 + Col;

  Number := 9;  Sum := 0;
  for I := 1 to Number do begin
    S[I] := FNum + (I - 1) * Inc;
    Sum  := Sum + S[I];
  end;
  MNum := Sum div 3;
  Permute (Number, S);
  Writeln ('MAGIC NUMBER = ', MNum);
end.


{ -- **********   Alternate solution for 3.10  ********** }
program Thr10T94;
{ -- This program produces a 3 x 3 magic square. }
  type
    String9  = Array [1..3, 1..3] of Integer;
  var
    I, J, FNum, Inc, MNum, Sum:         Integer;
    Num1, Num2, Row1, Col1, Row2, Col2: Integer;
    S: String9;

procedure FillRow;
begin
{ -- Determine missing row element from the other two. }
  for I := 1 to 3 do begin
    if (S[I, 1] = 0) and (S[I, 2] > 0) and (S[I, 3] > 0) then
      S[I, 1] := MNum - S[I, 2] - S[I, 3];
    if (S[I, 1] > 0) and (S[I, 2] = 0) and (S[I, 3] > 0) then
      S[I, 2] := MNum - S[I, 1] - S[I, 3];
    if (S[I, 1] > 0) and (S[I, 2] > 0) and (S[I, 3] = 0) then
      S[I, 3] := MNum - S[I, 1] - S[I, 2];
  end;
end;

procedure FillCol;
{ -- Determine missing column element from the other two. }
begin
  for J := 1 to 3 do begin
   if (S[1, J] = 0) and (S[2, J] > 0) and (S[3, J] > 0) then
     S[1, J] := MNum - S[2, J] - S[3, J];
   if (S[1, J] > 0) and (S[2, J] = 0) and (S[3, J] > 0) then
     S[2, J] := MNum - S[1, J] - S[3, J];
```

```
   if (S[1, J] > 0) and (S[2, J] > 0) and (S[3, J] = 0) then
     S[3, J] := MNum - S[1, J] - S[2, J];
  end;
end;


begin
  Write ('Enter first number: ');  Readln(FNum);
  Write ('Enter increment: ');     Readln(Inc);

  Write ('Enter number: ');    Readln (Num1);
  Write ('Enter row, col: ');  Readln (Row1, Col1);
  Write ('Enter number: ');    Readln (Num2);
  Write ('Enter row, col: ');  Readln (Row2, Col2);

  Sum := 0;
  for I := 1 to 3 do
    for J := 1 to 3 do begin
      S[I, J] := 0;
      Sum  := Sum + FNum + ((I-1) * 3 + (J-1)) * Inc;
  end;
  MNum := Sum div 3;  { -- Magic Number }

  S[Row1, Col1] := Num1;
  S[Row2, Col2] := Num2;
  S[2, 2] := Sum div 9;   { -- Middle number is always Sum / 9 }
  { -- Compute the element on the opposite ends of the 2 Nums. }
  S[4-Row1, 4-Col1] := MNum - S[2, 2] - S[Row1, Col1];
  S[4-Row2, 4-Col2] := MNum - S[2, 2] - S[Row2, Col2];

  FillRow;
  FillCol;
  FillRow;

  { -- Display the magic square and magic number. }
  for I := 1 to 3 do begin
    for J := 1 to 3 do
      Write (S[I, J] : 3);
    Writeln;
  end;
  Writeln;
  Writeln ('MAGIC NUMBER = ', MNum);
end.
```