```
{ -- 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.
```