FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95

1.1 Write a program to display the following lines, each beginning at the left most column of the screen:

```
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
```

1.2 Comments are used to explain sections of programming code. A comment statement is implemented a little differently in BASIC, Pascal, C, and C++. In BASIC, comments usually begin with an apostrophe (') but may also begin with the characters REM (short for REMARK). In Pascal, comments are usually delimited by curly braces ({ and }) but can be delimited by a parenthesis-asterisk and an asterisk-parenthesis { (* and *) }. In C, comments are only delimited by a slash-asterisk and an asterisk-slash (/* and */). In C++, comments usually begin with a double slash (//) but can be delimited by the same characters that the language C uses.

Write a program to accept as input a generic comment and then display it in the format commonly used by each of the languages mentioned. Example:

```
 INPUT: Enter comment: THIS PROGRAM WILL GENERATE COMMENTS

OUTPUT: BASIC: ' THIS PROGRAM WILL GENERATE COMMENTS
        PASCAL: { THIS PROGRAM WILL GENERATE COMMENTS }
        C: /* THIS PROGRAM WILL GENERATE COMMENTS */
        C++: // THIS PROGRAM WILL GENERATE COMMENTS
```

1.3 The C/C++ language includes two unary operators that are not found in other programming languages: the increment (++) and the decrement (--) operators. Increment adds one to a variable, and the decrement subtracts one from a variable; Thus, the language C++ is known as an incremental improvement over the language C.

Write a program to accept as input an integer N along with either the increment or decrement operator, and then display the new value for N: If N=-3 then N++ makes N=-2, and N-- makes N=-4. Examples:

```
 INPUT: Enter N: 5                  INPUT: Enter N: 20
        Enter operator: ++                 Enter operator: --
OUTPUT: 6                          OUTPUT: 19
```

1.4 Rounding off to three decimal places usually uses the rule that if the ten thousandth's digit is 5 or larger then round up, and if it is 4 or smaller then round down. The number 5 is called the break point. Write a program to accept as input a break point as any digit from 1 to 9 and accept as input a decimal number less than 10, and then display the number rounded to three decimal places. Examples:

```
 INPUT: Enter break point: 7      INPUT: Enter break point: 7
        Enter number: 1.3766             Enter number: 9.47979
OUTPUT: 1.376                    OUTPUT: 9.480
```

1.5 Time Sharing Option/Extensions (TSO/E) and Interactive System Productivity Facility (ISPF/PDF) are very useful for accessing a mainframe computer. REXX and CLIST are the two interpretative programming languages that can issue TSO and ISPF/PDF commands. REXX (REstructured eXtended eXecutor) and CLIST (Command LIST) code can be executed in the foreground within the MVS operating system. Since both languages are interpretative, each line is interpreted, and then executed, one line at a time, starting with the first line. The programs can be executed without being compiled and link-edited. BASIC is also commonly used as an interpretative programming language. Most other languages must be compiled and link-edited to create executable program modules.

CLIST is basically a command processor with limited programming functionality. REXX is a full application development programming language. REXX's structure and syntax closely resembles the language Pascal and to a lesser degree, PL/I. PL/I utilizes features from both COBOL (a business language) and FORTRAN (a scientific/mathematical language). REXX became the standard procedure language for IBM's System Application Architecture (SAA) in 1987, which means REXX is implemented across IBM's product line and used under several operating system environments. Starting with TSO/E Version 2, SYSPROC libraries can hold both CLIST and REXX programs, but the operating system needs to know what kind of program to execute. Therefore all REXX code stored in SYSPROC must start with a comment line that contains the word REXX on the first line.

Write a program to accept as input the first line of code as a comment and to display whether the operating system would interpret the program as a CLIST or a REXX. All comments begin with /* and end with */. If the four contiguous characters, REXX, appear somewhere in the first line of the comment then the operating system would interpret the code as a REXX program, otherwise it is seen as a CLIST program. Examples:

```
 INPUT: Enter comment: /* RESTRUCTURED EXTENDED EXECUTOR */
OUTPUT: CLIST

 INPUT: Enter comment: /* MY FIRST REXX-PROGRAM */
OUTPUT: REXX
```

1.6 In order to do well in a computer contest such as the FHSCC'95, a team must be quick in writing small programs. Any one of the following languages may be used in the contest: BASIC, Pascal, C, or C++. Beginners All-Purpose Symbolic Instruction Code (BASIC) was developed in the 1960's at Dartmouth College and was originally used on mainframes before becoming the most widely used programming language for microcomputers. BASIC is very easy to learn and the new versions contain powerful programming statements. Niklaus Wirth developed a new language in the early 1970's and named it after the 17th century mathematician, Blaise Pascal. Pascal is a highly structured and easy-to-maintain programming language that allows programmers to produce efficient programs. Dennis Ritchie at the Bell Telephone Laboratories developed a new language in order to design their UNIX operating system in 1972: C. Although C uses more special operators and symbols than most other languages (making it cryptic), C is the most popular professional programming language for microcomputers, enabling programmers to produce highly efficient code. AT&T's Bell Laboratories created the first C++ language in the 1980's to improve the way C works. C++ is an efficient language that has better C commands and the capability of using object-oriented programming (OOP).

Although BASIC, Pascal, and C/C++ are all good programming languages to use in this contest, programming in BASIC tends to be quicker to code since most variables do not need to be declared nor initialized. Pascal and C/C++ require that all variables be defined before they are used, whereas BASIC does not require a variable to be defined before it is used and automatically initializes all numeric variables to 0 and all strings to null. For example, to write the BASIC code "SUM = SUM + I + J" equivalently in either Pascal or C/C++ requires an additional appearance of the three variables by defining them before they are used (i.e. "var SUM, I, J: real;", or "float SUM, I, J;", respectively). An additional statement is required in Pascal to initialize SUM to zero (i.e. "SUM = 0;"), whereas C/C++ can initialize at the same time a variable is defined (i.e. "float SUM=0, I, J;"). Moreover, C/C++ can initialize variables "I" and "J" to a non-zero number at the same time they are defined, whereas both BASIC and Pascal must have a separate statement that assigns "I" and "J" to a non-zero number.

Write a program to accept as input the number of numeric variables used in a program and the number of those variables that need to be initialized and of those the number that need to be initialized specifically to zero, and then display the least number of times the variables must appear in declarations or statements before they may be used in a program for each of the three languages.

```
    INPUT: Enter number of variables: 6
           Enter number initialized: 4
           Enter number initialized to 0: 3
   OUTPUT: BASIC = 1
           PASCAL = 10
           C/C++ = 6
```

1.7 Frank is called a "toolie" in the Configuration Management group on the CBSS project because he develops tools (programs using REXX) to assist the CM technicians in their daily tasks on the mainframe computer. Frank's programs need to read in files (data set names) and assimilate the last part of the data set name. A "qualified data set" name has all the information necessary to locate the data set via the system catalog and consists of two or more unqualified data set names connected by periods. These unqualified data set names (or qualifiers) consist of one to eight characters, the first being alphabetic or national (@, $, #) and the remaining characters must be alphameric, national, or a hyphen. The qualified data set names cannot be longer than 44 characters including the periods. The high-level qualifier is the first qualifier (or unqualified name) in the data set name.

Write a program to parse and display the last qualifier on a qualified data set name that is given as input. Examples:

```
 INPUT: Enter data set name: DT10005.REXX.EXEC
OUTPUT: EXEC

 INPUT: Enter data set name: G001246.CBSFCONV.SPUFI.IN.CBST012
OUTPUT: CBST012
```

1.8 Write a program to first accept as input a positive integer N less than 10. Next, the program is to accept as input N real numbers between -9999.9999 and 9999.9999, inclusive, and then display these N real numbers in reverse order on the screen, exactly as they were input. Example:

```
 INPUT: Enter N: 5
        Enter #: 1.23
        Enter #: -123.40
        Enter #: 999.9999
        Enter #: 0.0
        Enter #: -1234.4567

OUTPUT: -1234.4567
        0.0
        999.9999
        -123.40
        1.23
```

1.9 Write a program to display a large 'X' on the screen made up of letter X's. The program is to accept as input an odd number (between 3 and 15, inclusive) representing the number of X's to be displayed on each main diagonal. The top-left and bottom-left 'X' must appear in column 1 on the screen. Example:

```
     INPUT: Enter number of X's: 7

    OUTPUT: (Screen clears and the following appears)
            X     X
             X   X
              X X
               X
              X X
             X   X
            X     X
```

1.10 GTE is an environmentally conscious corporate citizen. One of GTE's most environmentally impacting areas is the use of paper for phone bills. Considering this, GTE has decided to print bills on both sides of the paper (duplex) and save the equivalent of 3,000 80-foot trees per year. Printing on both sides reduces the number of pages in each bill by approximately 30 percent. In addition, this saves GTE a million dollars a month in the cost of forms and reduced postage charges. Assume that there are only six bill pages (front and back) in the first ounce of a bill due to the weight of the return envelope and an average of two inserts. Assume that each ounce after the first can have 9 pages (front and back). The long distance carriers have required that their bill pages not start on the back of any other carriers pages and that no other carrier appear on the back of their pages. GTE pays 23 1/3 cents per ounce for postage. Fractional ounces are paid using the next whole ounce. Tim has worked hard on developing this technology and would like to know how much GTE is saving so that management can adjust the postage budgets accordingly.

Write a program to accept as input the number of printed sides in a bill, and of those, the number of sides that will have a blank back side, and then determine the postage savings for that bill (in the format ###.## CENTS), compared to the cost of postage for a bill where all pages are printed on one side only. For example, if 50 sides are printed and 7 of those are single sided then (50-7)/2= 22 pages are double sided with duplex printing (total of 29 pages) as opposed to having 50 pages with single sided printing. Examples:

```
     INPUT: Enter # of printed sides: 50
            Enter # of single sided pages: 7
    OUTPUT:  46.67 CENTS SAVED


     INPUT: Enter # of printed sides: 62
            Enter # of single sided pages: 10
    OUTPUT:  70.00 CENTS SAVED
```

2.1 Write a program to find that integral solution of (X,Y) for AX+BY=C for which X is as small a positive integer as possible and A, B, and C are input as integers between -100 and 100, inclusive. Examples:

```
 INPUT: Enter A, B, C: 13, 21, 1
OUTPUT: (13,-8)

 INPUT: Enter A, B, C: 17, -19, 1
OUTPUT: (9,8)
```

2.2 Write a program to verify a "part number" by validating the check digit appearing in the units position. The computation of the check digit involves multiplying by 2 every other digit of the original number, starting with the first, and adding these values and the remaining digits of the number together. (Do not consider the right-most digit as a part of the number.) The units digit of the result obtained is then subtracted from 9 to obtain the check digit, which was not used in the computation. Using the part number in the first example below, 126547 becomes (1*2 + 6*2 + 4*2) + 2 + 5 = 29, ==> 9 - 9 = 0, and 0 does not match 7. The program is to accept as input a string of at most 20 digits, and display whether the part number is OKAY or in ERROR. If it is in ERROR, then display the correct check digit. Examples:

```
 INPUT: Enter part number: 126547
OUTPUT: ERROR -- CHECK DIGIT SHOULD BE 0

 INPUT: Enter part number: 1265400
OUTPUT: OKAY
```

2.3 Since computer education is the future of this nation, an imaginary millionaire's club would like to reward the efforts of the winners of the computer contest with 13 million dollars. Each of the winning teams will be awarded one of the following amounts: $1, $13, $169, $2197, $28561, $371293, and $4826809 (each a power of 13). If each prize (that is awarded) is given at most 9 times and the sum of all the awards total 13 million dollars, then write a program to determine how many of each prize will be awarded to the computer teams. Display the answer in the format below, where the symbol {#} represents a digit from 0 to 9:

```
OUTPUT: $1 = #
        $13 = #
        $169 = #
        $2197 = #
        $28561 = #
        $371293 = #
        $4826809 = #
```

2.4 Directory Assistance operators can be very beneficial to a person who does not know the entire phone number that they would like to call. A customer in the 813 area code may make up to three local Directory Assistance Calls (DAC) during each monthly billing period at no cost; Thereafter, each DAC in the local area costs 25 cents each. DAC's made within the 813 area code that are considered long distance are charged 25 cents per call. DAC's made to other area codes within Florida (i.e. 305, 407, 904) are charged 40 cents per call. DAC's made to places beyond Florida within the U.S. are charged 65 cents per call. International DAC's cost $3.00.

Write a program to first accept as input the number of DAC's made during the monthly billing period for a customer whose phone is in the 813 area code. The program is to then accept as input each DAC number and then display the cost associated with all these calls in the format ##.## DOLLARS. Each input will consist of at most 11 consecutive digits for a DAC. The DAC variations are shown below:

1411	Local Directory Assistance
1 + 813 + 555-1212	Numbers within area code
1 + area code + 555-1212 ..	Numbers outside area code
00	International calls

Example:

```
INPUT: Enter number of DAC's: 9
       Enter DAC: 1411
       Enter DAC: 1411
       Enter DAC: 18135551212
       Enter DAC: 00
       Enter DAC: 14075551212
       Enter DAC: 12025551212
       Enter DAC: 1411
       Enter DAC: 1411
       Enter DAC: 1411

OUTPUT:  4.80 DOLLARS
```

2.5 The new book entitled: FLORIDA HIGH SCHOOLS COMPUTING COMPETITION: PROBLEMS, JUDGING CRITERIA, BASIC SOLUTIONS, PASCAL SOLUTIONS: 1985 - 1994, by Douglas E. Woolley, contains 300 intriguing programming contest items and solutions. This 778 page book is a tool for enhancing computer programming skills and a preparation guide for those competing in contests such as this. The book is divided into four parts: Problems, Judging Criteria, BASIC solutions, and Pascal solutions. Each part is divided into 10 chapters corresponding to the years 1985 to 1994.

Write a program to display the heading of a page of the book given the page number as input. Assume that each chapter within a part has the same number of pages and that all pages are numbered consecutively and that the number of pages in each part is 180, 140, 200, and 260, respectively. Every even numbered page has in its heading the page number followed by 2 spaces and the title of the book: FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994. Every odd numbered page has in its heading the abbreviated title (FHSCC) followed by a space, followed by an apostrophe and the last two digits of the year of the contest and a space, followed by one of the four parts of the book: PROBLEMS, JUDGING CRITERIA, BASIC SOLUTIONS, or PASCAL SOLUTIONS; followed by two spaces and the page number. Examples:

```
 INPUT: Enter page number: 8
OUTPUT: 8  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994


 INPUT: Enter page number: 51
OUTPUT: FHSCC '87 PROBLEMS  51


 INPUT: Enter page number: 181
OUTPUT: FHSCC '85 JUDGING CRITERIA  181


 INPUT: Enter page number: 755
OUTPUT: FHSCC '94 PASCAL SOLUTIONS  755


 INPUT: Enter page number: 444
OUTPUT: 444  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994
```

2.6 The Internal Revenue Service (IRS) has compiled a table of "Estimated Preparation Time" to complete and file Form 1040 and its schedules:

				Copying,
				assembling,
		Learning about	Preparing	and sending
Form	Recordkeeping	the law/form	the form	form to IRS
Form 1040	3 hr., 8 min.	2 hr., 53 min.	4 hr., 41 min.	53 min.
Sch. A	2 hr., 32 min.	26 min.	1 hr., 10 min.	27 min.
Sch. B	33 min.	8 min.	17 min.	20 min.
Sch. C	6 hr., 26 min.	1 hr., 10 min.	2 hr., 5 min.	35 min.
Sch. D	51 min.	42 min.	1 hr., 1 min.	41 min.
Sch. E	2 hr., 52 min.	1 hr., 7 min.	1 hr., 16 min.	35 min.

Write a program to enter at most 6 unique forms and display the total ESTIMATED PREPARATION TIME to complete all stages of the forms designated. Valid input will consist of: 1040, A, B, C, D, or E; Input is terminated by an invalid entry. Output will be displayed with the minutes between 0 and 59 inclusive. Examples:

```
 INPUT: Enter form: D
        Enter form: 1040
        Enter form: NO

OUTPUT: 14 HR., 50 MIN.


 INPUT: Enter form: B
        Enter form: F

OUTPUT: 1 HR., 18 MIN.
```

2.7 At GTE there are many investment incentives for employees, such as the 401K investment plan and the guaranteed stock returns.

The 401K is a plan where an employee can contribute up to 16% of his/her income into investment funds. The company will match each dollar with 75 cents, of the first 6% contributed, with a Company-Matching Contribution, credited to the employee's account at the end of the year. These combined funds have returned a yearly interest rate, ranging from 6% to 29%, that is added to the employee's account.

Under the terms of the stock purchase plan, an employee can purchase one share of common stock for each full $100 of their annual basic rate of pay up to a maximum of 750 shares. The company sells each share to the employee at a guaranteed 85% of the "Average Market Price." If stock prices are higher at the end of the year than at the beginning, then the employee could earn more than 15% but never less.

Write a program that will allow an employee to see how much he can profit by investing. Input will first consist of the yearly salary and the percent of the 401K that will be contributed. Next, the program is to display the number of shares the employee can purchase, and then accept as input the number of shares that are bought, followed by the closing market price of a share (which will be greater than the starting value). Assume that stock prices (or "Average Market Price") start at $32.00 per share at the beginning of the year (thus employees purchase a share at $32.00 * 0.85), and 14% is the return on the combined employee/company contributions to the 401K. Output must consist of the company contribution, the 401K return, the gain in stock, and the total of these three gains, all in the form #####.##. Examples:

```
 INPUT: Enter salary: 32080
        Enter 401K %: 16
OUTPUT: YOU CAN PURCHASE UP TO 320 SHARES
 INPUT: Enter number of shares: 320
        Enter end of year price: 35.00
OUTPUT: COMPANY CONTRIBUTION:  1443.60
                 401K RETURN:   920.70
                  STOCK GAIN:  2496.00
                  TOTAL GAIN:  4860.30


 INPUT: Enter salary: 54321
        Enter 401K %: 4
OUTPUT: YOU CAN PURCHASE UP TO 543 SHARES
 INPUT: Enter number of shares: 100
        Enter end of year price: 33.25
OUTPUT: COMPANY CONTRIBUTION:  1629.63
                 401K RETURN:   532.35
                  STOCK GAIN:   605.00
                  TOTAL GAIN:  2766.98
```

2.8 Write a program to replicate the following pattern on the screen, given as input the number of spiral loops to display (less than 6) and the letter to be used in the first spiral. Each succeeding spiral will use the next letter in the alphabet (except Z is followed by A). After accepting input, the screen clears and the first character of the spiral is centered on the screen. Example:

```
 INPUT: Enter number of spiral loops: 4
        Enter first letter: Y

OUTPUT: (Screen clears and the following is centered)

                  B
                  B BBBBBBBBBBBBBBB
                  B A             B
                  B A AAAAAAAAAAA B
                  B A Z         A B
                  B A Z ZZZZZZZ A B
                  B A Z Y     Z A B
                  B A Z Y YYY Z A B
                  B A Z Y   Y Z A B
                  B A Z YYYYY Z A B
                  B A Z       Z A B
                  B A ZZZZZZZZZ A B
                  B A           A B
                  B AAAAAAAAAAAAA B
                  B               B
                  BBBBBBBBBBBBBBBBB
```

2.9 Write a program to display all possible moves for the Queen on an empty chess board. The program is to first accept as input the coordinates of the Queen (column A-H followed by row 1-8), and then clears the screen and displays the board layout in the upper left corner of the screen. A Queen may move horizontally, vertically, or diagonally. The letter 'Q' marks the position of the Queen and asterisks mark the possible moves for the Queen. Example:

```
 INPUT: Enter column and row: C4

OUTPUT: (Screen clears and the following appears)
        8     *       *
        7     *     *
        6 *   *   *
        5   * * *
        4 * * Q * * * * *
        3   * * *
        2 *   *   *
        1     *     *
          A B C D E F G H
```

2.10 During a pre-election poll, information was collected concerning: sex, age, race, income, and party to vote for. Each set of these categories will continue to be input until an 'E' is entered for sex. Valid inputs for sex are: M for Male, F for Female, or E to End; for race: W for White, or O for Other; and for party: D for Democratic, or R for Republican. Write a program to tabulate the data collected and generate a report showing percentages of each category among the Democratic and Republican parties as shown below. The column headings DEMOCRATIC and REPUBLICAN begin in columns 33 and 45 respectively. Percentages are displayed in the format ###.#. Example:

```
     INPUT: Enter sex: M
            Enter age: 23
            Enter race: O
            Enter income: 19000
            Enter party: R

            Enter sex: F
            Enter age: 67
            Enter race: W
            Enter income: 34000
            Enter party: R

            Enter sex: M
            Enter age: 51
            Enter race: W
            Enter income: 56000
            Enter party: D

            Enter sex: E

    OUTPUT:                                 DEMOCRATIC  REPUBLICAN
            MALE                                  33.3        33.3
            FEMALE                                 0.0        33.3

            50 AND BELOW                           0.0        33.3
            OVER 50                               33.3        33.3

            WHITE                                 33.3        33.3
            OTHERS                                 0.0        33.3

            ABOVE $25000                          33.3        33.3
            $25000 AND BELOW                       0.0        33.3

            WHITE MALE OVER 50 AND ABOVE $25000   33.3         0.0
            OTHER                                  0.0        66.7
```

3.1 As the first quarter of the year approaches, many people are working on their tax return. Write a program to determine how much money an individual tax payer will either pay to the IRS or receive back from the IRS.

The program is to first accept as input the adjusted gross income of a single person and the amount of itemized deductions. If the deductions are greater than the standard deduction of $3,800, then subtract the itemized amount from the adjusted gross income; otherwise subtract $3,800 from the adjusted gross income. Subtract an additional $2,450 (for one claimed exemption) to produce the taxable income.

Each year the IRS produces a tax table corresponding to taxable income less than $100,000. For incomes that exceed $100,000, a tax rate schedule is used instead. Even though the tax rate schedule is not used for income less than $100,000, this formula will be used in your program for all income levels and will produce amounts within $8 of an actual tax table look-up for incomes less than $100,000. Tax is computed as follows:

```
15%   of the first $22,750 plus
28%   of the amount over  $22,750 up to  $55,100 plus
31%   of the amount over  $55,100 up to $115,000 plus
36%   of the amount over $115,000 up to $250,000 plus
39.6% of the amount over $250,000
```

The program is also to accept as input the amount of federal income tax withheld from the person. If this amount is less than the computed tax, then the difference is owed to the IRS; otherwise the IRS owes the difference. Display the amount owed in the format: ######.## DOLLARS. Examples:

```
 INPUT: Enter adjusted gross income: 32140.65
        Enter itemized deductions: 4758.00
        Enter federal income tax withheld: 4062.00

OUTPUT:     38.36 DOLLARS WILL BE REFUNDED TO YOU


 INPUT: Enter adjusted gross income: 306250.00
        Enter itemized deductions: 3456.00
        Enter federal income tax withheld: 11222.00

OUTPUT:  88217.50 DOLLARS YOU OWE
```

3.2 GTE has become the largest U.S.-based local exchange carrier, with more than 22 million access lines worldwide. The GTE phone company uses a complex computer application called CBSS to bill its valued customers. In a simplified fashion, GTE charges customers a long distance rate determined by the length of a phone call in minutes, the time of day the call was placed, and to where the call was placed. Assuming that the rate chart below is in effect, write a program to produce a simplified phone bill for a customer given a series of phone calls input. Each phone call consists of the length in MINutes, and the time in the form: HH:MM AM DAY
where AM could also be PM, and DAY is the first 3 letters of the name (i.e. MON, TUE, WED, THU, FRI, SAT, SUN). The last call recorded is indicated by entering a 0 for the next prompt of MIN. If the customer's bill is over $20, then give a 20% discount, otherwise give a 0% discount. Display the times without the leading zero as shown below, and compute the total charges, the discount, and the charges minus the discount. Assume that the phone bill is for BOB SMITH who calls from his home at 813-555-1234 and always makes calls to the same long distance number. Note: 11am is followed by 12pm, then 1pm; 11pm is followed by 12am, then 1am. The rates are as follows (the first rate is for the first minute, the second rate is for all subsequent minutes):

Weekday Rates (Mon - Fri)		Weekend Rates (11pm Fri - 7:59am Mon)
8am - 4:59pm	.28 / .21	.14 / .11
5pm - 10:59pm	.21 / .16	(except 5pm Sun - 10:59pm Sun)
11pm - 7:59am	.14 / .11	.21 / .16

Example:

```
     INPUT: Enter MIN: 1
            Enter time: 07:56 AM  MON
            Enter MIN: 25
            Enter time: 12:01 PM  THU
            Enter MIN: 35
            Enter time: 03:15 PM  SAT
            Enter MIN: 84
            Enter time: 11:59 AM  FRI
            Enter MIN: 20
            Enter time: 10:09 AM  WED
            Enter MIN: 0
```

```
    OUTPUT:  BOB SMITH  (813) 555-1234

              TIME OF DAY  MIN.  CHARGE
             7:56 AM  MON    1     0.14
            12:01 PM  THU   25     5.32
             3:15 PM  SAT   35     3.88
            11:59 AM  FRI   84    17.71
            10:09 AM  WED   20     4.27

            TOTAL CHARGES         31.32
            DISCOUNT               6.26
            CHARGES - DISCOUNT    25.06
```

3.3 Write a program to simulate a baseball game of 9 innings. The standard baseball rules apply, but the bottom of the 9th inning is always played. Pitchers randomly throw strikes 40% of the time and the batters never swing at the ball. If 4 balls are thrown before 3 strikes are thrown, the batter walks to first base. When 4 batters from one team walk in one inning, 1 run is earned. Each batter that walks thereafter in the same inning earns a run for the team. 3 strikes make 1 out, and after 3 outs the next team bats. Because the program is random, executions will differ slightly. Examples:

```
                1  2  3  4  5  6  7  8  9  SCORE
                ---------------------------------
       TEAM A ! 0  0  0  3  0  3  0  0  0 !   6
       TEAM B ! 1  0  0  2  0  0  0  2  0 !   5

       TOTAL # OF STRIKES: 235
       TOTAL # OF BALLS: 343
       TOTAL # OF WALKS: 77
       TOTAL # OF STRIKE OUTS: 54


                1  2  3  4  5  6  7  8  9  SCORE
                ---------------------------------
       TEAM A ! 0  1  0  0  1  1  0  0  0 !   3
       TEAM B ! 0  0  2  0  0  2  2  0  3 !   9

       TOTAL # OF STRIKES: 251
       TOTAL # OF BALLS: 385
       TOTAL # OF WALKS: 88
       TOTAL # OF STRIKE OUTS: 54
```

3.4 Write a program that will accept up to 8 distinct letters in a string and output a list of all possible subsets of the list. Each subset will be listed alphabetically within the subset and in ascending order amongst the other subsets. The output must have as many complete subsets on a 50-character line as possible, with one space separating each subset. On the line after the last set of subsets, the total number of subsets must be displayed. Example:

```
  INPUT: Enter letters: CABFE
 OUTPUT: {} {A} {AB} {ABC} {ABCE} {ABCEF} {ABCF} {ABE}
         {ABEF} {ABF} {AC} {ACE} {ACEF} {ACF} {AE} {AEF}
         {AF} {B} {BC} {BCE} {BCEF} {BCF} {BE} {BEF} {BF}
         {C} {CE} {CEF} {CF} {E} {EF} {F}
         TOTAL SUBSETS = 32


  INPUT: Enter letters: ZYX
 OUTPUT: {} {X} {XY} {XYZ} {XZ} {Y} {YZ} {Z}
         TOTAL SUBSETS = 8
```

3.5 Write a program for Mr. Gauss to accurately and efficiently compute the sum of the integers from 1 to N, where N is input as a positive integer having less than 40 digits. Examples:

```
     INPUT: Enter number N: 100000000000000000000000000000
    OUTPUT:
    5000000000000000000000000000050000000000000000000000000000


     INPUT: Enter number N: 999999999999999999999999999999999
    OUTPUT:
499999999999999999999999999999995000000000000000000000000000000000
```

3.6 Write a program to input several lines of BASIC code and display the final values of all the variables used. All statements are executed in the order input and are of the form:

```
  variable = <variable/constant> [<operator> <variable/constant>]

                - variable is any single letter
                - constant is any single digit
                - operator is +, -, *, or /.
```

The last line of the program will be indicated by 'END'. All variables used on the right side of the equal sign {=} will have been previously assigned a value. All variables are to be displayed in the order that they are used in the program and all the values displayed will be integers. Examples:

```
     INPUT: Enter line: A=5
            Enter line: B=9
            Enter line: A=B+7
            Enter line: B=A-B
            Enter line: END

    OUTPUT: A=16
            B=7


     INPUT: Enter line: J=2
            Enter line: E=J*3
            Enter line: S=E
            Enter line: U=7*7
            Enter line: S=J-5
            Enter line: J=2+E
            Enter line: E=E/2
            Enter line: END

    OUTPUT: J=8
            E=3
            S=-3
            U=49
```

3.7 Write a program to find all sets of three 3-digit primes composed of the digits 1 through 9 such that their sum consists of four distinct digits in order of magnitude. Output must be of the following format:

```
### + ### + ### = ####
```

where ### represents the primes displayed in increasing order, and #### represents their sum. The seven sets of primes are to be displayed in order of magnitude by the first prime and then the second prime (if two sets have the same first prime). Two of the seven solutions are displayed below. Example:

```
149 + 257 + 863 = 1269
### + ### + ### = ####
### + ### + ### = ####
241 + 367 + 859 = 1467
### + ### + ### = ####
### + ### + ### = ####
### + ### + ### = ####
```

3.8 Write a program to clear the screen and display a runner's digital stop-watch time in block numbers given the minutes and seconds as input. The time must increment by one second approximately every second: No more than 15 seconds and no less than 7 seconds are to be displayed every 10 actual seconds. Program terminates upon pressing any key. All times are to be displayed in the upper-left corner of the screen in block numbers 4 asterisks wide and 5 asterisks high:

```
   ****     *  ****  ****  *  *  ****  *     ****  ****  ****
   *  *     *     *     *  *  *  *     *        *  *  *  *  *
   *  *     *  ****  ****  ****  ****  ****     *  ****  ****
   *  *     *  *        *     *     *  *  *     *  *  *     *
   ****     *  ****  ****     *  ****  ****     *  ****     *
```

Example:

INPUT: Enter MM:SS: **09:58**

OUTPUT: (Screen is cleared and the time is displayed in the upper-left corner of screen)

```
****  ****     ****  ****
*  *  *  *  *  *     *  *
*  *  ****     ****  ****
*  *     *  *     *  *  *
****     *     ****  ****
```

(approximately 1 second later the following appears)

```
****  ****     ****  ****
*  *  *  *  *  *     *  *
*  *  ****     ****  ****
*  *     *  *     *     *
****     *     ****     *
```

(approximately 1 second later the following appears)

```
   *  ****     ****  ****
   *  *  *  *  *  *  *  *
   *  *  *     *  *  *  *
   *  *  *  *  *  *  *  *
   *  ****     ****  ****
```

(approximately 1 second later the following appears)

```
   *  ****     ****     *
   *  *  *  *  *  *     *
   *  *  *     *  *     *
   *  *  *  *  *  *     *
   *  ****     ****     *
```

INPUT: (press any key)

OUTPUT: (program terminates)

3.9 GTE Data Services was incorporated on Oct. 25, 1967 and has recently restructured its four regional Information Processing Centers (IPCs) into three Information Control Centers (ICCs) in Tampa, Florida; Sacramento, California; and Fort Wayne, Indiana. Each of the buildings located in these areas have many different rooms and work cubicles.

Write a program to calculate the area of a room in the shape of a polygon with perpendicular corners, given a series of movements describing its shape. After the program accepts the number of vertical and horizontal sides in the room, it then accepts a list of successive direction-distance pairs, starting from an arbitrary corner. Directions will be U, D, R, and L to indicate Up, Down, Right, and Left respectively. Each direction will be followed by a distance in feet, less than 25. Each room described will have at most 10 corners and will have both a length and a width less than 25 feet. The first example uses a polygon room with the shape and dimensions of:

```
             24
  ***********************
  *                     *
4 *                     *
  ********              * 7
     8   *              *
       3 *              *
         ****************
               16
```

Examples:

```
     INPUT: Enter number of sides: 6
            Enter movement: U3
            Enter movement: L8
            Enter movement: U4
            Enter movement: R24
            Enter movement: D7
            Enter movement: L16

    OUTPUT: AREA = 144 SQUARE FEET


     INPUT: Enter number of sides: 10
            Enter movement: R8
            Enter movement: U2
            Enter movement: R6
            Enter movement: D10
            Enter movement: L10
            Enter movement: U3
            Enter movement: L9
            Enter movement: U7
            Enter movement: R5
            Enter movement: D2

    OUTPUT: AREA = 147 SQUARE FEET
```

3.10 Jim is the Distribution Coordinator for GTEDS CBSS Project. One aspect of his job is to assign which week of the year to create a new version of a library of data sets (files). The following is his base criteria:

- Each Version of a library spends 12 weeks in a test area and is named R1VvvL01, where vv is the Version number;
- Immediately following this test phase, the library is moved to Production for 6 weeks;
- 1 week before a library is moved to Production, a new Pre-Production test library is created and is functional for 6 weeks; This library is called R1VvvL88;
- 6 weeks after a Version enters a test area, the next Version of the library goes to the other test area; This Version follows the same time frames as listed above and is named similar to the previous Version, except that this Version is one greater in number;
- There are 2 test areas, and they alternate Versions of the library; all even Versions are in Test 1; all odd in Test 2.

Write a program to display the time relationships of these library versions by a horizontal graph. Input will consist of (a) a version number; (b) the week number that it goes to a test area; (c) the first week and the number of weeks to display on the graph (each less than 50). The program is to clear the screen and then display each week number vertically, starting in column 10. The versions are to be displayed in the order that they are created, each beginning in column 1. The program must show the time a version is in a test area by displaying a 1 or a 2. Display the weeks that a version is in Production with a P. Display the weeks a version has a Pre-Production test area with an asterisk. Example:

```
     INPUT: Enter version #: 36
            Enter first week in test: 2
            Enter first week to display, # of weeks: 4, 34
    OUTPUT: (Screen clears and the following displays)
                     0000001111111111222222222233333333
                     4567890123456789012345678901234567

            R1V34L01 PPPP
            R1V35L01 2222PPPPPP
            R1V34L88 ***
            R1V36L01 1111111111PPPPPP
            R1V35L88    ******
            R1V37L01     222222222222PPPPPP
            R1V36L88          ******
            R1V38L01           111111111111PPPPPP
            R1V37L88                ******
            R1V39L01                 222222222222PPPPPP
            R1V38L88                      ******
            R1V40L01                       111111111111
            R1V39L88                            ******
            R1V41L01                             222222
            R1V40L88                                  *
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '96

1.1 FHSCC is an abbreviation for Florida High Schools Computing Competition. Write a program to accept as input a four-digit year between 1980 and 1996, inclusive, and append the last two digits of the year to the phrase: **FHSCC '**. Examples:

```
  INPUT: Enter year: 1996
 OUTPUT: FHSCC '96

  INPUT: Enter year: 1984
 OUTPUT: FHSCC '84
```

1.2 Write a program to tally the number of frequent flier miles that Doug earns if he flies to and from Caracas, Venezuela **X** times and stays at the Hilton each time and pays a total of **Y** dollars in phone calls. The distance of the flight is 1300 miles one-way. Each "stay" at the Hilton earns 500 miles, and each dollar spent on phone calls earns 5 miles. Total will be less than 32767. Examples:

```
  INPUT: Enter X: 3              INPUT: Enter X: 2
         Enter Y: 10                    Enter Y: 100
 OUTPUT: 9350                   OUTPUT: 6700
```

1.3 Write a program to print the middle letter or letters of a given word. If the word has an even number of letters, then there will be two middle letters. If the word has an odd number of letters, then there will be one middle letter. Examples:

```
  INPUT: Enter word: DOUG
 OUTPUT: OU

  INPUT: Enter word: WOOLLEY
 OUTPUT: L
```

1.4 Write a program to accept the two end coordinates (X,Y) of one of the diagonals of a rectangle in the Cartesian plane whose sides are parallel to the X or Y-axis. The program must then display the area and perimeter of the rectangle. Examples:

```
  INPUT: Enter coordinate 1: -1, 2
         Enter coordinate 2: 4, -2
 OUTPUT: AREA = 20
         PERIMETER = 18

  INPUT: Enter coordinate 1: 3, 1
         Enter coordinate 2: 0, 0
 OUTPUT: AREA = 3
         PERIMETER = 8
```

1.5 Write a program to code-break a given encrypted secret message made up of alphabetic characters and spaces. The decoder must translate the letters ABCDEFGHIJKLMNOPQRSTUVWXYZ into the corresponding letters ZYXWVUTSRQPONMLKJIHGFEDCBA, respectively. A space is decoded into a space. The encryption will not contain more than 40 characters. Example:

```
  INPUT: Enter encryption: UOLIRWZ SRTS HXSLLO
 OUTPUT: FLORIDA HIGH SCHOOL

  INPUT: Enter encryption: XLNKFGVI XLMGVHG
 OUTPUT: COMPUTER CONTEST
```

1.6 A nice hotel in Caracas, Venezuela has 26 floors above the ground floor. Write a program to accept as input the floors on which an elevator of this hotel stops consecutively, and determine the total number of floors that an elevator of this hotel touches and the number of unique floors that the elevator touches. The elevator always starts on the ground (floor 0) and ends on the ground (floor 0). Therefore, the elevator starts by touching the ground floor and ends touching the ground floor. In the first example below, the elevator touches floors 0,1,2,3,4,5 then floors 4,3, then floor 4, then floors 3,2,1,0 for a total of 13 floors. In the second example below, the elevator touches floors 0,1,2, then floors 3,4, then floors 3,2,1,0 for a total of 9 floors. Examples:

```
  INPUT: Enter floor: 5
         Enter floor: 3
         Enter floor: 4
         Enter floor: 0

 OUTPUT: TOTAL FLOORS TOUCHED = 13
         UNIQUE FLOORS TOUCHED = 6


  INPUT: Enter floor: 2
         Enter floor: 4
         Enter floor: 0

 OUTPUT: TOTAL FLOORS TOUCHED = 9
         UNIQUE FLOORS TOUCHED = 5


  INPUT: Enter floor: 20
         Enter floor: 5
         Enter floor: 24
         Enter floor: 10
         Enter floor: 0

 OUTPUT: TOTAL FLOORS TOUCHED = 79
         UNIQUE FLOORS TOUCHED = 25
```

1.7 Write a program to determine a person's ratios for buying a house and whether he qualifies for a mortgage if a mortgage company will not approve a loan with ratios over 33% / 38%. All amounts input are monthly tallies. To qualify for the loan, the first ratio must not exceed 33% and the second ratio must not exceed 38%. The first ratio computes the loan amount divided by the income. The second ratio computes the sum of the loan and other debts divided by the income. Display the first and second ratios in the format: RATIOS = ##.#% / ##.#%, where each ratio is rounded to the nearest tenth of a percent . Display if this person DOES QUALIFY or DOES NOT QUALIFY for the loan. Examples:

```
 INPUT: Enter amount of loan: 900
        Enter amount of debts: 200
        Enter amount of income: 2800

OUTPUT: RATIOS = 32.1% / 39.3%
        DOES NOT QUALIFY


 INPUT: Enter amount of loan: 1000
        Enter amount of debts: 170
        Enter amount of income: 3100

OUTPUT: RATIOS = 32.3% / 37.7%
        DOES QUALIFY
```

1.8 Write a program to convert numbers 1-10 to the English or Spanish word for the number. The program will first prompt for either the letter 'E' for English or 'S' for Spanish. Next, the program will accept as input a number between 1 and 10, inclusive, and display the name of the number in the requested language.

English: ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE TEN
Spanish: UNO DOS TRES CUATRO CINCO SEIS SIETE OCHO NUEVE DIEZ

```
 INPUT: Enter E or S: E
        Enter number: 4

OUTPUT: FOUR


 INPUT: Enter E or S: S
        Enter number: 10

OUTPUT: DIEZ


 INPUT: Enter E or S: S
        Enter number: 5

OUTPUT: CINCO
```

1.9 Write a program to form a cross with the word(s) input, given that the total number of characters input is an odd number and the word(s) are to intersect at the middle character. Example:

```
     INPUT: Enter word(s): THE CROSS
    OUTPUT:     T
                H
                E

            THE CROSS
                R
                O
                S
                S
```

1.10 Write a program to simulate the PRICE IS RIGHT game. Given an item's actual price, and unique price guesses from four contestants, determine which person comes the closest to the price without going over the actual cost. If every contestant goes over the price, then display the message "EVERYONE IS OVER". Examples:

```
     INPUT: Enter actual price: 425
            Enter guesses A, B, C, D: 300, 400, 500, 200
    OUTPUT: PERSON B

     INPUT: Enter actual price: 399
            Enter guesses A, B, C, D: 300, 400, 500, 301
    OUTPUT: PERSON D

     INPUT: Enter actual price: 299
            Enter guesses A, B, C, D: 300, 400, 500, 301
    OUTPUT: EVERYONE IS OVER
```

2.1 Write a program that will emulate random dart throws. The dart scores possible are 0,2,4,5,10,20,50 with the probability of hitting any score being the same as any other. The object of the game is to accumulate a score of at least 100 points. The program will print the score of each dart throw, separated by a comma, until the sum of the scores totals 100 points or more. The program must then print the number of throws that achieved the score, followed by the total score achieved. Sample RANDOM runs:

```
OUTPUT: 2,4,20,4,10,0,5,20,4,2,50
          11 THROWS ACHIEVED SCORE OF 121

OUTPUT: 50,20,10,5,10,2,5
          7 THROWS ACHIEVED SCORE OF 102
```

2.2 Write a program to compress information and save space, given a string of data. Input will consist of a string of letters with one or more asterisks (representing spaces) between words. The program must display the string of data with multiple asterisks replaced by the number of asterisks being eliminated. If only one asterisk separates two words, the asterisk should not be replaced with the number 1 since space would not be conserved. Examples:

```
 INPUT: Enter string: WE*CONSERVE****SPACE**BY***COMPRESSION
OUTPUT: WE*CONSERVE4SPACE2BY3COMPRESSION

 INPUT: Enter string: THIS**SENTENCE*IS*****COMPRESSED
OUTPUT: THIS2SENTENCE*IS5COMPRESSED
```

2.3 Set "A" has the property that the product of any two elements (numbers in the set) is 1 less than a perfect square. The numbers 1, 3, and 8 are elements of this set, i.e. 1x3=(4-1), 1x8=(9-1), 3x8=(25-1). Write a program to find two other whole numbers less than 1000 that are also elements of set "A". Display the solutions in numerical order in the format shown below, where # represents a digit. Example output format:

```
OUTPUT: #
        ###
```

2.4 Write an efficient program to display the least common multiple of a set of integers from 1 to N, inclusive, where N is at most 30. The least common multiple is a positive integer that is evenly divisible by every integer in the set and will contain at most 13 digits. Examples:

```
 INPUT: Enter N: 6              INPUT: Enter N: 30
OUTPUT: 60                     OUTPUT: 2329089562800
```

2.5 Write a program to calculate a fractional value of three-letter words. The reciprocal value of a letter in the alphabet (A...Z) is defined as the reciprocal of the position of that letter in the alphabet (A=1/1, B=1/2, C=1/3 ... Z=1/26). The fractional value is the sum of the reciprocals of the value of each letter in that word. This value must be printed as a simplified fraction. In the first example below, CAB is 1/3 + 1/1 + 1/2 = 11/6. In the second example below, EAT is 1/5 + 1/1 + 1/20 = (4 + 20 + 1)/20 = 25/20 = 5/4. Examples:

```
 INPUT: Enter word: CAB
OUTPUT: 11/6

 INPUT: Enter word: EAT
OUTPUT: 5/4
```

2.6 The Fibonacci sequence has the property that each number (beyond the first two) is the sum of the previous two numbers (1,1,2,3,5,8,...). The first two numbers in the sequence are 1 and 1. The third number is the sum of 1 and 1, or 2. The fourth number is the sum of 1 and 2, or 3. The fifth number is the sum of 2 and 3, or 5, etc.

Write a program to accept as input a number, N, less than 10, and display the Nth prime within the Fibonacci sequence. The first prime number in the sequence is the number 2. Examples:

```
 INPUT: Enter N: 3
OUTPUT: 5

 INPUT: Enter N: 9
OUTPUT: 514229
```

2.7 GTE sorts its phone bills by postal code then by telephone number before the bills are printed. Write a program to accept as input a list of (at most 8) four-digit phone numbers followed by zip code, and display the order in which the phone numbers will be printed. Input will be terminated by the entry **0000, 00000**. All other entries will not have any leading zeros. Example:

```
 INPUT: Enter phone #, zip: 1796, 33647
        Enter phone #, zip: 1521, 33555
        Enter phone #, zip: 2001, 33647
        Enter phone #, zip: 1400, 33647
        Enter phone #, zip: 1621, 33555
        Enter phone #, zip: 0000, 00000

OUTPUT: 1521
        1621
        1400
        1796
        2001
```

2.8 Write a program to display the findings of a statistical test on a set of random letters. Given a string of letters, display the number of runs of letters that are in the first half of the alphabet (A,B,C,D,E,F,G,H,I,J,K,L,M), and the number of runs of letters that are in the second half of the alphabet (N,O,P,Q,R,S,T, U,V,W,X,Y,Z). A run is a continuous group of elements in the same category. A string of FLANOMZUGODISGOODF consists of the runs: FLA, NO, M, ZU, G, O, DI, S, G, OO, DF. Examples:

```
  INPUT: Enter letters: FLANOMZUGODISGOODF
 OUTPUT: RUNS IN 1ST HALF = 6
         RUNS IN 2ND HALF = 5

  INPUT: Enter letters: XPQJESUSISLORDQPY
 OUTPUT: RUNS IN 1ST HALF = 4
         RUNS IN 2ND HALF = 5
```

2.9 Write a program to reverse the order of letters in each word of a given string unless the word is a palindrome (a word that is spelled the same forward and backward). If the word is a palindrome, then replace each letter with a question mark (?). Each word is separated by a single space. Examples:

```
  INPUT: Enter string: HOW GOOD IT IS FOR REMER
 OUTPUT: WOH DOOG TI SI ROF ?????

  INPUT: Enter string: OTTO CAME UP WITH THE WORD ROADAOR
 OUTPUT: ???? EMAC PU HTIW EHT DROW ???????
```

2.10 Write a program to determine the day of the week that a given date falls by using the following three tables and algorithm:

MONTH NUMBERS		
January	1	(if leap year then use 0)
February	4	(if leap year then use 3)
March	4	
April	0	
May	2	
June	5	
July	0	
August	3	
September	6	
October	1	
November	4	
December	6	

CENTURY NUMBERS	
1753 to 1799	add 4
1800 to 1899	add 2
1900 to 1999	add 0
2000 to 2099	add 6
2100 to 2199	add 4

DAY NUMBERS	
Saturday	= 0
Sunday	= 1
Monday	= 2
Tuesday	= 3
Wednesday	= 4
Thursday	= 5
Friday	= 6

Sum the following five derived numbers:

1) The last two digits of the year.
2) The whole quotient of this number divided by 4.
3) The MONTH NUMBER associated with the input month.
4) The day of the month for the input date.
5) The CENTURY NUMBER associated with the input year.

Next, divide the sum of the five numbers above by 7 and compare the remainder with the DAY NUMBERS to obtain the corresponding day.

Input will be three numbers corresponding to the month, day, and year. Note that a leap year is divisible by 4, except for those years also divisible by 100; but, if the year is also divisible by 400 then it is still a leap year. In the first example below, the algorithm uses the input date of August 4, 1856, and divides the sum of (56 + 14 + 3 + 4 + 2) by 7, which equals 11 remainder 2. The number 2 corresponds to Monday, so August 4, 1856 was a Monday. Examples:

INPUT: Enter month, day, year: **8, 4, 1856**
OUTPUT: **MONDAY**

INPUT: Enter month, day, year: **9, 27, 1990**
OUTPUT: **THURSDAY**

INPUT: Enter month, day, year: **2, 1, 1996**
OUTPUT: **THURSDAY**

3.1 Write a program to display the appearance of a 3-dimensional book with the two title lines centered vertically on the spine. The spine's height is to be 2 characters more than the longest title line. The book's width is to be 5 characters. In the process of centering the shorter title line, if there is an extra space it should succeed the title line. Title lines will not exceed 17 characters. The left side of the book must be in column 1 and the top of the book must be on row 1 of the screen. Examples:

```
     INPUT: Enter title 1: WHERE WE
            Enter title 2: STAND
    OUTPUT: (Screen clears, left side in column 1, top in row 1)
                /---/!
               /   / !
              /   /  !
             /   /   !
            !---!    !
            !  W!    !
            !S H!    !
            !T E!    !
            !A R!    !
            !N E!    !
            !D  !   /
            !  W!  /
            !  E! /
            !---!/


     INPUT: Enter title 1: EVIDENCE THAT
            Enter title 2: DEMANDS A VERDICT
    OUTPUT: (Screen clears, left side in column 1, top in row 1)
                /---/!
               /   / !
              /   /  !
             /   /   !
            !---!    !
            !D  !    !
            !E  !    !
            !M E!    !
            !A V!    !
            !N I!    !
            !D D!    !
            !S E!    !
            !  N!    !
            !A C!    !
            !  E!    !
            !V  !    !
            !E T!    !
            !R H!    !
            !D A!    !
            !I T!   /
            !C  !  /
            !T  ! /
            !---!/
```

3.2 Write a program to produce a prime factors tree for a given number. As seen in the example below, the symbols {/} and {\} appear beneath the largest non-prime factors' left and right, respectively. The smallest prime factors on the bottom-left of each {/} are to be displayed in ascending order on each succeeding pair of lines. The larger factor on each line is the dividend of the number appearing above it and the prime on its left. The input number will have less than 10 prime factors and none of the factors will have more than four digits. The program must clear the screen and display the input number beginning in column 6. Examples:

```
 INPUT: Enter number: 100

OUTPUT: (Screen is cleared, and the following appears)
             100
            /   \
           2     50
                /  \
               2    25
                   /  \
                  5    5
```

```
 INPUT: Enter number: 1716

OUTPUT: (Screen is cleared, and the following appears)
             1716
            /    \
           2      858
                 /   \
                2     429
                     /   \
                    3     143
                         /   \
                       11     13
```

3.3 Write a program to simulate a "base four" calculator that accepts expressions involving +, -, and unsigned integers in base 4. Each integer input will be less than 6 digits long and the result must be displayed in base 4. No expression will contain more than 40 characters. Examples:

```
 INPUT: Enter base 4 expression: 1230+23-3210-123+10
OUTPUT: -2010

 INPUT: Enter base 4 expression: 123-12-12+23-321+333
OUTPUT: 200
```

3.4 Write a program to calculate the daily amount of money that a contractor makes for working between a given time frame. Input will consist of the hourly rate of pay in dollars for contractors who work more than 50 percent of their time during normal working hours (7 AM - 5 PM). If at least 50 percent of the work is done outside of normal hours then the pay rate is to be increased by a "shift differential" of 10 percent to be applied for all the hours worked. Contractors will work less than 12 hours per day. The start and finish time will be input in the form HH:MM and followed by either AM or PM. Note: 11:59AM is followed by 12:00PM, and 11:59PM is followed by 12:00AM. Output must be of the format $DDD.CC with no leading zeros in dollars and have trailing zeros in cents. Examples:

```
 INPUT: Enter pay/hour: 9.05
        Enter start time: 08:00AM
        Enter finish time: 07:00PM
OUTPUT: $ 99.55

 INPUT: Enter pay/hour: 20.00
        Enter start time: 12:15PM
        Enter finish time: 09:45PM
OUTPUT: $209.00
```

3.5 On a panel of 16 buttons (4 rows of 4 buttons), each button must be pressed once and in the correct order. The final button to be pressed is always marked 0F for Final. The number of moves and the direction is marked on each button. 1R means one move right; 2D means two moves down; 3U means three moves up; 1L means one move left.

Write a program to print the first button that you must press (and its position) that will lead you to press every other button and finish at the final button 0F. In the first example below, pressing the 3L button leads to 1D, 3R, 2U, 1U, 2L, 3D, 1R, 3U, 2L, 1D, 2R, 1L, 1D, 1R, 0F. Examples:

```
 INPUT: Enter row: 1D 3D 2L 2L
        Enter row: 2R 1D 1L 1U
        Enter row: 1D 1R 0F 3L
         Enter row: 3R 1R 3U 2U

OUTPUT: FIRST BUTTON = 3L
        AT ROW = 3, COL = 4


 INPUT: Enter row: 0F 3D 3D 2L
        Enter row: 2R 1D 1U 2L
        Enter row: 2U 1L 1R 1U
         Enter row: 2U 1L 1U 3U

OUTPUT: FIRST BUTTON = 3U
        AT ROW = 4, COL = 4
```

3.6 A magic square is a matrix of distinct numbers with the same number of rows as columns, and the sum of the numbers in each row, column, and diagonal is equal to the same total (the magic number). There are a number of general methods for generating magic squares with an odd "order" (number of rows and columns). La Loubre invented the staircase method:

1) Start with a number in the top middle square.

2) The next number (incremented by a constant) is placed diagonally up and right in the next box of the array. If the number would be placed outside of the array, then the number is moved to another spot in the array according to these two rules: If the top row is exceeded, then it is placed in the bottom row; if the right-most column is exceeded, then it is placed in the first column.

3) If the square is already occupied while trying to place the number in the array, then the number is placed in the square that is immediately below the original number. If the bottom row is exceeded then the number is placed in the top row with the same column.

4) Continue to place numbers in the magic square by repeating steps 2 and 3 until all squares have been populated.

Write a program to display a magic square using the staircase algorithm, given as input an odd "order" for the magic square (at most 13), the first positive integer to use, and the positive integral increment between each successive number. In the magic square, each number is to be right justified within a four-character column. Begin the output by displaying the magic number (the sum of all the cells for a column, for a row, for a diagonal). Examples:

```
 INPUT: Enter order, first number, increment: 3, 2, 3

OUTPUT: MAGIC NUMBER = 42
          23   2  17
           8  14  20
          11  26   5


 INPUT: Enter order, first number, increment: 7, 90, 2

OUTPUT: MAGIC NUMBER = 966
         148 166 184  90 108 126 144
         164 182 102 106 124 142 146
         180 100 104 122 140 158 162
          98 116 120 138 156 160 178
         114 118 136 154 172 176  96
         130 134 152 170 174  94 112
         132 150 168 186  92 110 128
```

3.7 A magic square is a matrix of distinct numbers with the same number of rows as columns and the sum of the numbers in each row, column, and diagonal is equal to the same total (the magic number). All magic squares with an odd "order" (number of rows and columns) can be generated using La Loubre's staircase method. General methods are still being explored for generating magic squares with an even "order" (number of rows and columns). Magic squares whose order is a multiple of four can be constructed by a particular method. However, the most difficult magic square to produce is one with an order of 6. The easiest method used to construct a 6 by 6 magic square is as follows:

1) Divide the 6 by 6 square into four 3 by 3 squares.

2) Using the "staircase" method to generate 3 by 3 magic squares, place the first nine numbers in the upper left 3 by 3 square, place the next nine numbers in the lower right-hand 3 by 3 square, place the next nine numbers in the upper right-hand 3 by 3 square, place the last nine numbers in the lower left-hand 3 by 3 square.

3) Transpose the three cells in positions (1,1), (2,2), (3,1) with those cells in positions (4,1), (5,2), and (6,1), respectively, where each position is designated by (Row, Column).

Write a program to display the 6 by 6 magic square using the staircase algorithm, given as input the first positive integer to use and the positive increment between each successive number. Each number is to be right justified within a four-character column. Begin the output by displaying the magic number (the sum of all the cells for a column, for a row, and for a diagonal).

For the example below, a 6 by 6 magic square starting with the number 1 and each successive number incremented by 1 would look like this after steps 1 and 2:

```
   8   1   6     26  19  24
   3   5   7     21  23  25
   4   9   2     22  27  20

  35  28  33     17  10  15
  30  32  34     12  14  16
  31  36  29     13  18  11
```

The final magic square is displayed below. Example:

```
     INPUT: Enter first number, increment: 1, 1

    OUTPUT: MAGIC NUMBER = 111
              35   1   6  26  19  24
               3  32   7  21  23  25
              31   9   2  22  27  20
               8  28  33  17  10  15
              30   5  34  12  14  16
               4  36  29  13  18  11
```

3.8 Write a program to display a pie graph on the screen using asterisks and the letters A, D, and N.

Input will be 3 percentages to divide the circle corresponding to 3 options on a survey: Agree, Disagree, or Neutral.

Output will be a circle of radius 10 characters. The circle is then partitioned by 3 line segments of asterisks stemming from the center. The first percentage entered (those that Agree) is represented by a proportional region of the circle enclosed by two segments: One segment is drawn from the center to the top of the circle, and another segment is drawn from the center to a point on the circle, enclosing a clock-wise region. Another segment is drawn from the center to the circle so that the next area clock-wise represents the percentage that disagrees. The third region clock-wise represents the percentage that is neutral. After the user presses a key, the 3 regions are filled with either A's, D's, or N's, corresponding to its region. Although your output should look very similar to the judging criteria, minor variations will be accepted. After pressing a key to fill the regions, all regions must be at least 90% filled. No letters may replace any of the asterisks. Example:

INPUT: Enter 3 percentages: **64, 11, 25**

```
OUTPUT:          *******           OUTPUT:          *******
               **   *   **                        **NNN*AAA**
              *     *     *                      *NNNNN*AAAAA*
             *      *      *                    *NNNNNN*AAAAAA*
            *       *       *                  *NNNNNNN*AAAAAAA*
           *        *        *                *NNNNNNNN*AAAAAAAA*
           *        *        *                *NNNNNNNN*AAAAAAAA*
          *         *         *              *NNNNNNNNN*AAAAAAAAA*
          *         *         *              *NNNNNNNNN*AAAAAAAAA*
          *         *         *              *NNNNNNNNN*AAAAAAAAA*
          ***********         *              ***********AAAAAAAAA*
          *       **          *              *DDDDDDD**AAAAAAAAAA*
          *      *            *              *DDDDDD*AAAAAAAAAAAA*
          *    **             *              *DDDD**AAAAAAAAAAAAA*
           *  *              *                *DD*AAAAAAAAAAAAAA*
           *  *              *                *DD*AAAAAAAAAAAAAA*
            **              *                  **AAAAAAAAAAAAAA*
             **            *                     **AAAAAAAAAAAA*
              *           *                       *AAAAAAAAAAA*
               **       **                         **AAAAAAA**
                 *******                             *******
```

INPUT: (press any key)

3.9 Write a program to produce THE UNIQUE ORDER of execution for jobs (that run Billing system programs) given a set of dependencies between the jobs. The program will first prompt for the number of dependencies (less than 8) that will be entered. Each input line of dependencies will consist of a 2-character job name that must finish before the second 2-character job, separated by a space. The output line must contain THE UNIQUE ORDERING of the jobs, all on one line, that will satisfy the dependencies. Examples:

```
 INPUT: Enter number of dependencies: 5
         Enter dependency: OA OU
        Enter dependency: OA OJ
        Enter dependency: OA OE
        Enter dependency: OJ OE
        Enter dependency: OE OU
OUTPUT: JOBS MUST BE RUN IN THIS ORDER: OA OJ OE OU

 INPUT: Enter number of dependencies: 6
         Enter dependency: BK 5M
        Enter dependency: BE BK
        Enter dependency: BM BN
        Enter dependency: 5M BN
        Enter dependency: BK BM
        Enter dependency: BM 5M
OUTPUT: JOBS MUST BE RUN IN THIS ORDER: BE BK BM 5M BN
```

3.10 The digits 123456789 can be rearranged to form a nine-digit perfect square with unique digits. For example, swapping 7 with 8, 6 with 9, 4 with 8, 3 with 7, 2 with 4, and 1 with 8, forms the perfect square 847159236 (the square of 29106). This square is formed by making six exchanges:

```
swap 7 with 8        123456879
swap 6 with 9         123459876
swap 4 with 8        123859476
swap 3 with 7        127859436
swap 2 with 4        147859236
swap 1 with 8        847159236
```

Write a program to find the nine-digit perfect square that requires the fewest exchanges of pairs of digits from the original 123456789 number. Display the square with its square root followed by the number of exchanges required to form the square. The format of the output must be two lines as follows in the first example with # representing a digit. The second example output shows what the output would be like IF the fewest number of exchanges is actually 6, but it is fewer. Example outputs:

```
OUTPUT: ######### IS THE SQUARE OF #####
        AND WAS FORMED BY EXCHANGING # PAIRS OF DIGITS

OUTPUT: 847159236 IS THE SQUARE OF 29106
        AND WAS FORMED BY EXCHANGING 6 PAIRS OF DIGITS
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
JUDGING CRITERIA

```
1.1  RUN PROGRAM:

    OUTPUT: FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
            59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
            FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
            59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
            FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
            59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
            FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
            59' NOITITEPMOC GNITUPMOC SLOOHCS HGIH ADIROLF
```

```
1.2  INPUT: Enter comment: COMMENTS ARE GENERATED IN THIS PROGRAM

    OUTPUT: BASIC: ' COMMENTS ARE GENERATED IN THIS PROGRAM
            PASCAL: { COMMENTS ARE GENERATED IN THIS PROGRAM }
            C: /* COMMENTS ARE GENERATED IN THIS PROGRAM */
            C++: // COMMENTS ARE GENARTED IN THIS PROGRAM
```

```
1.3  INPUT: Enter N: -15                INPUT: Enter N: 99
            Enter operator: ++                 Enter operator: --
    OUTPUT: -14                        OUTPUT: 98
```

```
1.4  INPUT: Enter break point: 3        INPUT: Enter break point: 3
            Enter number: 6.54321              Enter number: 7.65432
    OUTPUT: 6.543                      OUTPUT: 7.655


     INPUT: Enter break point: 9        INPUT: Enter break point: 9
            Enter number: 5.6788               Enter number: 6.78991
    OUTPUT: 5.678                      OUTPUT: 6.790
```

```
1.5  INPUT: Enter comment: /* COMMAND LIST PROGRAM */
    OUTPUT: CLIST

     INPUT: Enter comment: /* REXX */
    OUTPUT: REXX

     INPUT: Enter comment: /* THIS IS A 1-POINT-REXX PROGRAM */
    OUTPUT: REXX
```

```
1.6  INPUT: Enter number of variables: 15
            Enter number initialized: 9
            Enter number initialized to 0: 5
    OUTPUT: BASIC = 4
            PASCAL = 24
            C/C++ = 15

     INPUT: Enter number of variables: 10
            Enter number initialized: 2
            Enter number initialized to 0: 2
    OUTPUT: BASIC = 0
            PASCAL = 12
            C/C++ = 10
```

```
1.7  INPUT: Enter data set name: TTGTCBS.DOCLIB.PROJECT.SPEC
    OUTPUT: SPEC

     INPUT: Enter data set name: MYUSERID.DATASET
    OUTPUT: DATASET
```

```
1.8  INPUT: Enter N: 6
            Enter #: 9.1234
            Enter #: 10.500
            Enter #: -3.4
            Enter #: 7777.22
            Enter #: 0.0632
            Enter #: -234.0

    OUTPUT: -234.0
            0.0632
            7777.22
            -3.4
            10.500
            9.1234


     INPUT: Enter N: 2
            Enter #: 100.05
            Enter #: -3.500

    OUTPUT: -3.500
            100.05
```

```
1.9  INPUT: Enter number of X's: 13

    OUTPUT: (Screen clears and the following appears)
            X           X
             X         X
              X       X
               X     X
                X   X
                 X X
                  X
                 X X
                X   X
               X     X
              X       X
             X         X
            X           X


     INPUT: Enter number of X's: 3

    OUTPUT: (Screen clears and the following appears)
            X X
             X
            X X
```

```
1.10 INPUT: Enter # of printed sides: 80
            Enter # of single sided pages: 9
    OUTPUT:  93.33 CENTS SAVED


     INPUT: Enter # of printed sides: 300
            Enter # of single sided pages: 20
    OUTPUT: 350.00 CENTS SAVED
```

```
2.1  INPUT: Enter A, B, C: 17, 23, 2
    OUTPUT: (15,-11)

     INPUT: Enter A, B, C: 2, 3, 96
    OUTPUT: (3,30)

     INPUT: Enter A, B, C: -100, 99, 8
    OUTPUT: (91,92)
```

```
2.2  INPUT: Enter part number: 9876543210123456789
    OUTPUT: ERROR -- CHECK DIGIT SHOULD BE 7

     INPUT: Enter part number: 246801357964
    OUTPUT: OKAY
```

```
2.3  RUN PROGRAM:
     OUTPUT: $1 = 0
             $13 = 1
             $169 = 2
             $2197 = 2
             $28561 = 0
             $371293 = 9
             $4826809 = 2
```

```
2.4  INPUT: Enter number of DAC's: 11
            Enter DAC: 18135551212
            Enter DAC: 14075551212
            Enter DAC: 00
            Enter DAC: 1411
            Enter DAC: 00
            Enter DAC: 1411
            Enter DAC: 19045551212
            Enter DAC: 1411
            Enter DAC: 1411
            Enter DAC: 12125551212
            Enter DAC: 1411
    OUTPUT:  8.20 DOLLARS


     INPUT: Enter number of DAC's: 2
            Enter DAC: 12195551212
            Enter DAC: 1411
    OUTPUT:  0.65 DOLLARS
```

```
2.5
 INPUT: Enter page number: 320
OUTPUT: 320  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994


 INPUT: Enter page number: 341
OUTPUT: FHSCC '86 BASIC SOLUTIONS  341


 INPUT: Enter page number: 319
OUTPUT: FHSCC '94 JUDGING CRITERIA  319


 INPUT: Enter page number: 701
OUTPUT: FHSCC '91 PASCAL SOLUTIONS  701


 INPUT: Enter page number: 46
OUTPUT: 46  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION 1985 - 1994
```

```
2.6  INPUT: Enter form: A
            Enter form: B
            Enter form: C
            Enter form: D
            Enter form: E
            Enter form: 1040
            Enter form: F

    OUTPUT: 36 HR., 49 MIN.


     INPUT: Enter form: E
            Enter form: A
            Enter form: C
            Enter form: H

    OUTPUT: 20 HR., 41 MIN.
```

```
2.7  INPUT: Enter salary: 40100
            Enter 401K %: 10
    OUTPUT: YOU CAN PURCHASE UP TO 401 SHARES
     INPUT: Enter number of shares: 159
            Enter end of year price: 34.56
    OUTPUT: COMPANY CONTRIBUTION:  1804.50
                     401K RETURN:   814.03
                      STOCK GAIN:  1170.24
                      TOTAL GAIN:  3788.77


     INPUT: Enter salary: 50999
            Enter 401K %: 3
    OUTPUT: YOU CAN PURCHASE UP TO 509 SHARES
     INPUT: Enter number of shares: 500
            Enter end of year price: 36.00
    OUTPUT: COMPANY CONTRIBUTION:  1147.48
                     401K RETURN:   374.84
                      STOCK GAIN:  4400.00
                      TOTAL GAIN:  5922.32
```

```
2.8  INPUT: Enter number of spiral loops: 5
            Enter first letter: Z

    OUTPUT: (Screen clears and the following is centered)

                        D
                        D DDDDDDDDDDDDDDDDDDD
                        D C                 D
                        D C CCCCCCCCCCCCCCC D
                        D C B             C D
                        D C B BBBBBBBBBBB C D
                        D C B A         B C D
                        D C B A AAAAAAA B C D
                        D C B A Z     A B C D
                        D C B A Z ZZZ A B C D
                        D C B A Z   Z A B C D
                        D C B A ZZZZZ A B C D
                        D C B A       A B C D
                        D C B AAAAAAAAA B C D
                        D C B           B C D
                        D C BBBBBBBBBBBBB C D
                        D C               C D
                        D CCCCCCCCCCCCCCCCC D
                        D                   D
                        DDDDDDDDDDDDDDDDDDDDD
```

(INPUT/OUTPUT CONTINUED ON NEXT PAGE)

(INPUT/OUTPUT CONTINUED FOR 2.8)

```
     INPUT: Enter number of spiral loops: 1
            Enter first letter: F

    OUTPUT: (Screen clears and the following is centered)

                        F
                        F FFF
                        F   F
                        FFFFF
```

2.9

```
     INPUT: Enter column and row: F2

    OUTPUT: (Screens clears and the following appears)
            8           *
            7 *         *
            6   *       *
            5     *     *
            4       *   *   *
            3         * * *
            2 * * * * * Q * *
            1         * * *
              A B C D E F G H

     INPUT: Enter column and row: H8

    OUTPUT: (Screen clears and the following appears)
            8 * * * * * * * Q
            7             * *
            6           *   *
            5         *     *
            4       *       *
            3     *         *
            2   *           *
            1 *             *
              A B C D E F G H
```

```
2.10 INPUT: Enter sex: M
            Enter age: 23
            Enter race: O
            Enter income: 19000
            Enter party: R

            Enter sex: F
            Enter age: 67
            Enter race: W
            Enter income: 34000
            Enter party: R

            Enter sex: F
            Enter age: 47
            Enter race: W
            Enter income: 24000
            Enter party: D

            Enter sex: M
            Enter age: 51
            Enter race: W
            Enter income: 56000
            Enter party: D

            Enter sex: M
            Enter age: 50
            Enter race: O
            Enter income: 36000
            Enter party: D

            Enter sex: M
            Enter age: 51
            Enter race: W
            Enter income: 16000
            Enter party: R

            Enter sex: E

    OUTPUT:                                 DEMOCRATIC  REPUBLICAN
            MALE                                  33.3        33.3
            FEMALE                                16.7        16.7

            50 AND BELOW                          33.3        16.7
            OVER 50                               16.7        33.3

            WHITE                                 33.3        33.3
            OTHERS                                16.7        16.7

            ABOVE $25000                          33.3        16.7
            $25000 AND BELOW                      16.7        33.3

            WHITE MALE OVER 50 AND ABOVE $25000   16.7         0.0
            OTHER                                 33.3        50.0
```

```
3.1  INPUT: Enter adjusted gross income: 45678.90
            Enter itemized deductions: 3210.98
            Enter federal income tax withheld: 7000.00

    OUTPUT:   1082.59 DOLLARS YOU OWE


     INPUT: Enter adjusted gross income: 1234567.00
            Enter itemized deductions: 54321.00
            Enter federal income tax withheld: 555444.00

    OUTPUT: 108397.28 DOLLARS WILL BE REFUNDED TO YOU



3.2  INPUT: Enter MIN: 29
            Enter time: 08:50 AM  MON
            Enter MIN: 1
            Enter time: 05:50 PM  TUE
            Enter MIN: 2
            Enter time: 12:55 PM  WED
            Enter MIN: 16
            Enter time: 12:00 AM  THU
            Enter MIN: 67
            Enter time: 10:59 PM  FRI
            Enter MIN: 1
            Enter time: 12:00 PM  SAT
            Enter MIN: 30
            Enter time: 06:00 PM  SUN
            Enter MIN: 0

    OUTPUT:   BOB SMITH  (813) 555-1234

              TIME OF DAY  MIN.  CHARGE
             8:50 AM  MON   29     6.16
             5:50 PM  TUE    1     0.21
            12:55 PM  WED    2     0.49
            12:00 AM  THU   16     1.79
            10:59 PM  FRI   67    10.77
            12:00 PM  SAT    1     0.14
             6:00 PM  SUN   30     4.85

            TOTAL CHARGES         24.41
            DISCOUNT               4.88
            CHARGES - DISCOUNT    19.53
```

(INPUT/OUTPUT CONTINUED ON NEXT PAGE)

(INPUT/OUTPUT CONTINUED FOR 3.2)

```
     INPUT: Enter MIN: 11
            Enter time: 08:50 AM  SUN
            Enter MIN: 0

    OUTPUT:   BOB SMITH  (813) 555-1234

              TIME OF DAY  MIN.  CHARGE
             8:50 AM  SUN   11     1.24

            TOTAL CHARGES          1.24
            DISCOUNT               0.00
            CHARGES - DISCOUNT     1.24
```

3.3 RUN PROGRAM: (twice)

```
    OUTPUT: (Each run is random, but should be SIMILAR
             to the following baseball game results.
             Check that the score is correctly added.
             99% of the time this program will have:
             - each score in an inning less than 10,
             - total # of strikes between 211 and 280,
             - total # of balls between 290 and 470,
             - total # of walks between 69 and 111.)

                     1  2  3  4  5  6  7  8  9  SCORE
                    --------------------------------
            TEAM A ! 2  3  0  0  0  1  0  0  3 !   9
            TEAM B ! 2  0  1  2  3  0  0  0  2 !  10

            TOTAL # OF STRIKES: 247
            TOTAL # OF BALLS: 403
            TOTAL # OF WALKS: 92
            TOTAL # OF STRIKE OUTS: 54


                     1  2  3  4  5  6  7  8  9  SCORE
                    --------------------------------
            TEAM A ! 0  2  0  1  0  2  0  0  1 !   6
            TEAM B ! 0  0  0  0  0  0  0  1  0 !   1

            TOTAL # OF STRIKES: 239
            TOTAL # OF BALLS: 337
            TOTAL # OF WALKS: 76
            TOTAL # OF STRIKE OUTS: 54
```

```
3.4  INPUT: Enter letters: EGOAIMY

    OUTPUT: {} {A} {AE} {AEG} {AEGI} {AEGIM} {AEGIMO}
            {AEGIMOY} {AEGIMY} {AEGIO} {AEGIOY} {AEGIY}
            {AEGM} {AEGMO} {AEGMOY} {AEGMY} {AEGO} {AEGOY}
            {AEGY} {AEI} {AEIM} {AEIMO} {AEIMOY} {AEIMY}
            {AEIO} {AEIOY} {AEIY} {AEM} {AEMO} {AEMOY} {AEMY}
            {AEO} {AEOY} {AEY} {AG} {AGI} {AGIM} {AGIMO}
            {AGIMOY} {AGIMY} {AGIO} {AGIOY} {AGIY} {AGM}
            {AGMO} {AGMOY} {AGMY} {AGO} {AGOY} {AGY} {AI}
            {AIM} {AIMO} {AIMOY} {AIMY} {AIO} {AIOY} {AIY}
            {AM} {AMO} {AMOY} {AMY} {AO} {AOY} {AY} {E} {EG}
            {EGI} {EGIM} {EGIMO} {EGIMOY} {EGIMY} {EGIO}
            {EGIOY} {EGIY} {EGM} {EGMO} {EGMOY} {EGMY} {EGO}
            {EGOY} {EGY} {EI} {EIM} {EIMO} {EIMOY} {EIMY}
            {EIO} {EIOY} {EIY} {EM} {EMO} {EMOY} {EMY} {EO}
            {EOY} {EY} {G} {GI} {GIM} {GIMO} {GIMOY} {GIMY}
            {GIO} {GIOY} {GIY} {GM} {GMO} {GMOY} {GMY} {GO}
            {GOY} {GY} {I} {IM} {IMO} {IMOY} {IMY} {IO} {IOY}
            {IY} {M} {MO} {MOY} {MY} {O} {OY} {Y}
            TOTAL SUBSETS = 128


     INPUT: Enter letters: LORD

    OUTPUT: {} {D} {DL} {DLO} {DLOR} {DLR} {DO} {DOR} {DR}
            {L} {LO} {LOR} {LR} {O} {OR} {R}
            TOTAL SUBSETS = 16
```

```
3.5  INPUT: Enter N: 123456789012345678901234567890999
    OUTPUT:
762078937661941837524767578139155000992384766155479903221210545000


     INPUT: Enter N: 987654321098765432109876543210
    OUTPUT:
    987730528992531626293629019968318853833388126809944436823655
```

```
3.6  INPUT: Enter line: C=5
            Enter line: H=9-C
            Enter line: R=H*C
            Enter line: I=R/H
            Enter line: S=I
            Enter line: T=R+3
            Enter line: END

    OUTPUT: C=5
            H=4
            R=20
            I=5
            S=5
            T=23


     INPUT: Enter line: C=2
            Enter line: B=C*3
            Enter line: C=C-6
            Enter line: D=B
            Enter line: D=C/4
            Enter line: C=2*B
            Enter line: B=B+2
            Enter line: END

    OUTPUT: C=12
            B=8
            D=-1
```

```
3.7  RUN PROGRAM:

    OUTPUT: 149 + 257 + 863 = 1269
            149 + 263 + 857 = 1269
            239 + 587 + 641 = 1467
            241 + 367 + 859 = 1467
            257 + 419 + 683 = 1359
            263 + 419 + 587 = 1269
            283 + 457 + 619 = 1359
```

3.8 The screen will clear and display a runner's digital stop-watch time in block numbers given the minutes and seconds as input. The time must increment by one second approximately every second: No more than 15 seconds and no less than 7 seconds are to be displayed every 10 actual seconds. Program terminates upon pressing any key. All times are to be displayed in the upper-left corner of the screen in block numbers 4 asterisks wide and 5 asterisks long:

```
   ****     *  ****  ****  *  *  ****  *     ****  ****  ****
   *  *     *     *     *  *  *  *     *        *  *  *  *  *
   *  *     *  ****  ****  ****  ****  ****     *  ****  ****
   *  *     *  *        *     *     *  *  *     *  *  *     *
   ****     *  ****  ****     *  ****  ****     *  ****     *
```

```
     INPUT: Enter MM:SS: 03:58

    OUTPUT: (Screen is cleared and the time is displayed in
             the upper-left corner of screen)
            ****  ****     ****  ****
            *  *     *  *  *     *  *
            *  *  ****     ****  ****
            *  *     *  *     *  *  *
            ****  ****     ****  ****

            (approximately 1 second later the following appears)

            ****  ****     ****  ****
            *  *     *  *  *     *  *
            *  *  ****     ****  ****
            *  *     *  *     *     *
            ****  ****     ****     *

            (approximately 1 second later the following appears)

            ****  *  *     ****  ****
            *  *  *  *  *  *  *  *  *
            *  *  ****     *  *  *  *
            *  *     *  *  *  *  *  *
            ****     *     ****  ****

            (approximately 1 second later the following appears)

            ****  *  *     ****     *
            *  *  *  *  *  *  *     *
            *  *  ****     *  *     *
            *  *     *  *  *  *     *
            ****     *     ****     *

            (have the program display 19 more seconds then...)
     INPUT: (press any key)
    OUTPUT: (program terminates)
```

```
3.9  INPUT: Enter number of sides: 8
            Enter movement: L3
            Enter movement: U10
            Enter movement: R5
            Enter movement: U7
            Enter movement: R3
            Enter movement: D10
            Enter movement: L5
            Enter movement: D7

    OUTPUT: AREA = 66 SQUARE FEET


     INPUT: Enter number of sides: 10
            Enter movement: R5
            Enter movement: D12
            Enter movement: L5
            Enter movement: U2
            Enter movement: L2
            Enter movement: D2
            Enter movement: L6
            Enter movement: U5
            Enter movement: R8
            Enter movement: U7

    OUTPUT: AREA = 96 SQUARE FEET
```

```
3.10 INPUT: Enter version #: 47
            Enter first week in test: 8
            Enter first week to display, # of weeks: 3, 38

    OUTPUT: (Screen clears and the following displays)
                     00000001111111111222222222233333333334
                     34567890123456789012345678901234567890

            R1V44L01 PPPPP
            R1V45L01 22222PPPPPP
            R1V44L88 ****
            R1V46L01 11111111111PPPPPP
            R1V45L88     ******
            R1V47L01      222222222222PPPPPP
            R1V46L88          ******
            R1V48L01           111111111111PPPPPP
            R1V47L88                ******
            R1V49L01                 222222222222PPPPPP
            R1V48L88                      ******
            R1V50L01                       111111111111PPP
            R1V49L88                            ******
            R1V51L01                             222222222
            R1V50L88                                  ****
            R1V52L01                                   111


     INPUT: Enter version #: 36
            Enter first week in test: 2
            Enter first week to display, # of weeks: 25, 16

    OUTPUT: (Screen clears and the following displays)
                     2222233333333334
                     5678901234567890

            R1V37L01 P
            R1V38L01 1PPPPPP
            R1V39L01 2222222PPPPPP
            R1V38L88 ******
            R1V40L01  111111111111PPP
            R1V39L88      ******
            R1V41L01       222222222
            R1V40L88            ****
            R1V42L01             111
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '96
JUDGING CRITERIA

```
1.1  INPUT: Enter year: 1992
    OUTPUT: FHSCC '92

     INPUT: Enter year: 1980
    OUTPUT: FHSCC '80
```

```
1.2  INPUT: Enter X: 10
            Enter Y: 100
    OUTPUT: 31500

     INPUT: Enter X: 5
            Enter Y: 60
    OUTPUT: 15800
```

```
1.3  INPUT: Enter word: FLORIDA
    OUTPUT: R

     INPUT: Enter word: COMPUTER
    OUTPUT: PU

     INPUT: Enter word: COMPETITION
    OUTPUT: T
```

```
1.4  INPUT: Enter coordinate 1: 1, -5
            Enter coordinate 2: -5, -2
    OUTPUT: AREA = 18
            PERIMETER = 18

     INPUT: Enter coordinate 1: -3, 1
            Enter coordinate 2: 0, 12
    OUTPUT: AREA = 33
            PERIMETER = 28
```

```
1.5  INPUT: Enter encryption: GSV NBHGVIB GSZG LMXV DZH SRWWVM
    OUTPUT: THE MYSTERY THAT ONCE WAS HIDDEN

     INPUT: Enter encryption: UILN ZTVH GL TVMVIZGRLMH
    OUTPUT: FROM AGES TO GENERATIONS
```

```
1.6  INPUT: Enter floor: 5
            Enter floor: 7
            Enter floor: 4
            Enter floor: 18
            Enter floor: 3
            Enter floor: 0

    OUTPUT: TOTAL FLOORS TOUCHED = 43
            UNIQUE FLOORS TOUCHED = 19


     INPUT: Enter floor: 26
            Enter floor: 10
            Enter floor: 1
            Enter floor: 0

    OUTPUT: TOTAL FLOORS TOUCHED = 53
            UNIQUE FLOORS TOUCHED = 27


1.7  INPUT: Enter amount of loan: 398
            Enter amount of debts: 500
            Enter amount of income: 1230

    OUTPUT: RATIOS = 32.4% / 73.0%
            DOES NOT QUALIFY


     INPUT: Enter amount of loan: 1100
            Enter amount of debts: 150
            Enter amount of income: 3300

    OUTPUT: RATIOS = 33.3% / 37.9%
            DOES NOT QUALIFY


     INPUT: Enter amount of loan: 800
            Enter amount of debts: 200
            Enter amount of income: 3000

    OUTPUT: RATIOS = 26.7% / 33.3%
            DOES QUALIFY
```

```
1.8  INPUT: Enter E or S: E
             Enter number: 7
    OUTPUT: SEVEN

     INPUT: Enter E or S: S
            Enter number: 8
    OUTPUT: OCHO

     INPUT: Enter E or S: S
            Enter number: 1
    OUTPUT: UNO

     INPUT: Enter E or S: E
             Enter number: 3
    OUTPUT: THREE
```

```
1.9  INPUT: Enter word(s): HIGH SCHOOL
    OUTPUT:      H
                 I
                 G
                 H

            HIGH SCHOOL
                 C
                 H
                 O
                 O
                 L


     INPUT: Enter word(s): DOG
    OUTPUT:  D
            DOG
             G
```

```
1.10 INPUT: Enter actual price: 600
            Enter guesses A, B, C, D: 300, 400, 500, 200
    OUTPUT: PERSON C

     INPUT: Enter actual price: 399
            Enter guesses A, B, C, D: 600, 500, 400, 300
    OUTPUT: PERSON D

     INPUT: Enter actual price: 300
            Enter guesses A, B, C, D: 301, 402, 503, 604
    OUTPUT: EVERYONE IS OVER

     INPUT: Enter actual price: 425
            Enter guesses A, B, C, D: 425, 500, 400, 300
    OUTPUT: PERSON A
```

2.1 RUN PROGRAM:

(The program will emulate random dart throws. The dart scores possible are 0,2,4,5,10,20,50 with the probability of hitting any score being the same as any other. The object of the game is to accumulate a score of at least 100 points. The program will print the score of each dart throw, separated by a comma, until the sum of the scores totals 100 points or more. The program must then print the number of throws that achieved the score, followed by the total score achieved.)

- Ensure that all dart scores are only 0,2,4,5,10,20, or 50
- Ensure that the # of throws equals the # of scores shown above it
- Ensure that the final score is at least 100, and that the difference between the score and the last dart throw score is less than 100
- Ensure that the program "appears" random by running it several times

Sample RANDOM runs:

```
OUTPUT: 2,4,20,4,10,0,5,20,4,2,50
         11 THROWS ACHIEVED SCORE OF 121

OUTPUT: 50,20,10,5,10,2,5
         7 THROWS ACHIEVED SCORE OF 102
```

2.2
```
 INPUT: Enter string: FLORIDA*HIGH**SCHOOLS***COMPUTING
OUTPUT: FLORIDA*HIGH2SCHOOLS3COMPUTING

 INPUT: Enter string: COMPETITION*******FOR*****THIS*YEAR
OUTPUT: COMPETITION7FOR5THIS*YEAR
```

2.3 RUN PROGRAM:
```
OUTPUT: 0
        120
```

2.4
```
 INPUT: Enter N: 10              INPUT: Enter N: 28
OUTPUT: 2520                    OUTPUT: 80313433200
```

2.5
```
 INPUT: Enter word: FUN
OUTPUT: 2/7

 INPUT: Enter word: BAT
OUTPUT: 31/20
```

```
2.6  INPUT: Enter N: 7
    OUTPUT: 1597

     INPUT: Enter N: 8
    OUTPUT: 28657
```

```
2.7  INPUT: Enter phone #, zip: 1796, 33647
            Enter phone #, zip: 1521, 33555
            Enter phone #, zip: 2001, 33647
            Enter phone #, zip: 1400, 33647
            Enter phone #, zip: 1621, 33555
            Enter phone #, zip: 1555, 33647
            Enter phone #, zip: 0000, 00000

    OUTPUT: 1521
            1621
            1400
            1555
            1796
            2001


     INPUT: Enter phone #, zip: 3000, 33444
            Enter phone #, zip: 2000, 33555
            Enter phone #, zip: 2001, 33222
            Enter phone #, zip: 1000, 33444
            Enter phone #, zip: 4000, 33555
            Enter phone #, zip: 0000, 00000

    OUTPUT: 2001
            1000
            3000
            2000
            4000
```

```
2.8  INPUT: Enter letters: YETRULYTHEBIBLEISGODSWORD
    OUTPUT: RUNS IN 1ST HALF = 6
            RUNS IN 2ND HALF = 6

     INPUT: Enter letters: LORDJESUSISGODSSON
    OUTPUT: RUNS IN 1ST HALF = 5
            RUNS IN 2ND HALF = 5
```

```
2.9  INPUT: Enter string: WHAT DOES SIMIS MEAN
    OUTPUT: TAHW SEOD ????? NAEM

     INPUT: Enter string: OTTO GAVE A TOOT TO TOTO
    OUTPUT: ???? EVAG ? ???? OT OTOT
```

```
2.10 INPUT: Enter month, day, year: 2, 29, 1992
    OUTPUT: SATURDAY

     INPUT: Enter month, day, year: 10, 16, 1966
    OUTPUT: SUNDAY

     INPUT: Enter month, day, year: 2, 1, 1799
    OUTPUT: FRIDAY

     INPUT: Enter month, day, year: 1, 1, 2100
    OUTPUT: FRIDAY
```

```
3.1  INPUT: Enter title 1: THE HAPPIEST
            Enter title 2: PEOPLE ON EARTH

    OUTPUT: (Screen clears, left side in column 1, top in row 1)
                /---/!
               /   / !
              /   /  !
             /   /   !
            !---!    !
            !P  !    !
            !E T!    !
            !O H!    !
            !P E!    !
            !L  !    !
            !E H!    !
            !  A!    !
            !O P!    !
            !N P!    !
            !  I!    !
            !E E!    !
            !A S!    !
            !R T!   /
            !T  !  /
            !H  ! /
            !---!/


     INPUT: Enter title 1: MORE THAN A
            Enter title 2: CARPENTER

    OUTPUT: (Screen clears, left side in column 1, top in row 1)
                /---/!
               /   / !
              /   /  !
             /   /   !
            !---!    !
            !  M!    !
            !C O!    !
            !A R!    !
            !R E!    !
            !P  !    !
            !E T!    !
            !N H!    !
            !T A!    !
            !E N!   /
            !R  !  /
            !  A! /
            !---!/
```

```
3.2  INPUT: Enter number: 1130

    OUTPUT: (Screen is cleared, and the following appears)
                 1130
                /    \
               2      565
                     /   \
                    5     113


     INPUT: Enter number: 4864

    OUTPUT: (Screen is cleared, and the following appears)
                 4864
                /    \
               2      2432
                     /    \
                    2      1216
                          /    \
                         2      608
                               /   \
                              2     304
                                   /   \
                                  2     152
                                       /   \
                                      2     76
                                           /  \
                                          2    38
                                              /  \
                                             2    19
```

```
3.3  INPUT: Enter base 4 expression: 1230-23+3210+123-10
    OUTPUT: 11130

     INPUT: Enter base 4 expression: 12321-32101-21012+12321
    OUTPUT: -21211
```

```
3.4  INPUT: Enter pay/hour: 15.00
            Enter start time: 01:30AM
            Enter finish time: 12:10PM
    OUTPUT: $176.00

     INPUT: Enter pay/hour: 20.00
            Enter start time: 12:55PM
            Enter finish time: 12:25AM
    OUTPUT: $253.00

     INPUT: Enter pay/hour: 30.00
            Enter start time: 06:00AM
            Enter finish time: 05:25PM
    OUTPUT: $342.50
```

```
3.5  INPUT: Enter row: 1D 3D 2L 2L
            Enter row: 2R 1D 1L 1U
            Enter row: 1D 1R 1R 0F
             Enter row: 3R 1R 3U 2U
    OUTPUT: FIRST BUTTON = 1D
            AT ROW = 3, COL = 1


     INPUT: Enter row: 2R 2R 2D 0F
            Enter row: 1U 1U 1L 3L
            Enter row: 1R 1D 1R 3L
             Enter row: 3R 1R 2U 2U
    OUTPUT: FIRST BUTTON = 3R
            AT ROW = 4, COL = 1


3.6  INPUT: Enter order, first number, increment: 5, 9, 10
    OUTPUT: MAGIC NUMBER = 645
             169 239   9  79 149
             229  49  69 139 159
              39  59 129 199 219
              99 119 189 209  29
             109 179 249  19  89


     INPUT: Enter order, first number, increment: 7, 89, 2
    OUTPUT: MAGIC NUMBER = 959
             147 165 183  89 107 125 143
             163 181 101 105 123 141 145
             179  99 103 121 139 157 161
              97 115 119 137 155 159 177
             113 117 135 153 171 175  95
             129 133 151 169 173  93 111
             131 149 167 185  91 109 127



3.7  INPUT: Enter first number, increment: 2, 1
    OUTPUT: MAGIC NUMBER = 117
              36   2   7  27  20  25
               4  33   8  22  24  26
              32  10   3  23  28  21
               9  29  34  18  11  16
              31   6  35  13  15  17
               5  37  30  14  19  12

     INPUT: Enter first number, increment: 10, 25
    OUTPUT: MAGIC NUMBER = 2685
             860  10 135 635 460 585
              60 785 160 510 560 610
             760 210  35 535 660 485
             185 685 810 410 235 360
             735 110 835 285 335 385
              85 885 710 310 435 260
```

```
3.8  INPUT: Enter 3 percentages: 64, 11, 25
    OUTPUT:        *******          OUTPUT:        *******
                 **   *   **                     **NNN*AAA**
                *     *     *                   *NNNNN*AAAAA*
               *      *      *                 *NNNNNN*AAAAAA*
              *       *       *               *NNNNNNN*AAAAAAA*
             *        *        *             *NNNNNNNN*AAAAAAAA*
             *        *        *             *NNNNNNNN*AAAAAAAA*
            *         *         *           *NNNNNNNNN*AAAAAAAAA*
            *         *         *           *NNNNNNNNN*AAAAAAAAA*
            *         *         *           *NNNNNNNNN*AAAAAAAAA*
            ***********         *           ***********AAAAAAAAA*
            *       **          *           *DDDDDDD**AAAAAAAAAA*
            *      *            *           *DDDDDD*AAAAAAAAAAAA*
            *    **             *           *DDDD**AAAAAAAAAAAAA*
             *  *              *             *DD*AAAAAAAAAAAAAA*
             *  *              *             *DD*AAAAAAAAAAAAAA*
              **              *               **AAAAAAAAAAAAAA*
               **            *                  **AAAAAAAAAAAA*
                *           *                    *AAAAAAAAAAA*
                 **       **                      **AAAAAAA**
                   *******                          *******

    INPUT: (press any key)
```

Note: Although the output should look very similar to the judging criteria, minor variations will be accepted. After pressing a key to fill the regions, all regions must be at least 90% filled. No letters may replace any of the asterisks.

```
     INPUT: Enter 3 percentages: 25, 39, 36
    OUTPUT:        *******          OUTPUT:        *******
                 **   *   **                     **NNN*AAA**
                *     *     *                   *NNNNN*AAAAA*
               *      *      *                 *NNNNNN*AAAAAA*
              *       *       *               *NNNNNNN*AAAAAAA*
             *        *        *             *NNNNNNNN*AAAAAAAA*
             *        *        *             *NNNNNNNN*AAAAAAAA*
            *         *         *           *NNNNNNNNN*AAAAAAAAA*
            *         *         *           *NNNNNNNNN*AAAAAAAAA*
            *         *         *           *NNNNNNNNN*AAAAAAAAA*
            *         ***********           *NNNNNNNNN***********
            *       **          *           *NNNNNNN**DDDDDDDDDD*
            *      *            *           *NNNNNN*DDDDDDDDDDDD*
            *    **             *           *NNNN**DDDDDDDDDDDDD*
             *  *              *             *NN*DDDDDDDDDDDDDD*
             *  *              *             *NN*DDDDDDDDDDDDDD*
              **              *               **DDDDDDDDDDDDDD*
               **            *                  **DDDDDDDDDDDD*
                *           *                    *DDDDDDDDDDD*
                 **       **                      **DDDDDDD**
                   *******                          *******

     INPUT: (press any key)
```

```
3.9  INPUT: Enter number of dependencies: 5
            Enter dependency: PF PI
            Enter dependency: PA PF
            Enter dependency: PF PP
            Enter dependency: PI PP
            Enter dependency: PA PI
    OUTPUT: JOBS MUST BE RUN IN THIS ORDER: PA PF PI PP


     INPUT: Enter number of dependencies: 8
             Enter dependency: VS VD
            Enter dependency: V8 VI
            Enter dependency: VD VI
            Enter dependency: VA V7
            Enter dependency: V8 VS
            Enter dependency: V7 V8
            Enter dependency: VA VS
            Enter dependency: V7 VD
    OUTPUT: JOBS MUST BE RUN IN THIS ORDER: VA V7 V8 VS VD VI


3.10  RUN PROGRAM:

    OUTPUT: 523814769 IS THE SQUARE OF 22887
            AND WAS FORMED BY EXCHANGING 3 PAIRS OF DIGITS
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95
BASIC PROGRAM SOLUTIONS

```
'1.1
' This program displays title of contest forward and backward.
'
A$ = "FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95"
FOR I = 1 TO 4
    PRINT A$
    FOR J = LEN(A$) TO 1 STEP -1
      PRINT MID$(A$, J, 1);
    NEXT J
    PRINT
NEXT I
```

```
'1.2
' This program will generate comments in different languages.
'
INPUT "Enter comment:"; C$
PRINT "BASIC: ' "; C$
PRINT "PASCAL: { "; C$; " }"
PRINT "C: /* "; C$; " */"
PRINT "C++: // "; C$
```

```
'1.3
' This program either increments or decrements N by 1.
'
INPUT "Enter N:"; N
INPUT "Enter operator:"; OP$
IF OP$ = "++" THEN
  PRINT N + 1
ELSE          'OPerator is "--"
  PRINT N - 1
END IF
```

```
'1.4
' This program rounds to three decimal places by break point
'
INPUT "Enter break point:"; BP
INPUT "Enter number:"; NUM
ROUND = INT(NUM * 1000 + (10 - BP) / 10) / 1000
PRINT USING "#.###"; ROUND
```

```
'1.5
' This program will determine if a program is a REXX or a CLIST.
'
INPUT "Enter comment:"; C$
IF INSTR(C$, "REXX") > 0 THEN
  PRINT "REXX"
ELSE
  PRINT "CLIST"
END IF
```

```
'1.6
' This program displays the number of times variables appear.
'
INPUT "Enter number of variables:"; NUM
INPUT "Enter number initialized:"; INIT
INPUT "Enter number initialized to 0:"; INIT0
PRINT "BASIC ="; INIT - INIT0
PRINT "PASCAL ="; NUM + INIT
PRINT "C/C++ ="; NUM
```

```
'1.7
' This program displays the last qualifier of a data set name.
'
INPUT "Enter data set name"; DSN$
FOR I = LEN(DSN$) TO 1 STEP -1
  CH$ = MID$(DSN$, I, 1)
  IF CH$ = "." THEN
    PRINT LAST$: END
  ELSE
    LAST$ = CH$ + LAST$
  END IF
NEXT I
```

```
'1.8
' This program displays a set of real numbers in reverse order.
'
INPUT "Enter N:"; N
FOR I = 1 TO N
  INPUT "Enter #:"; A$(I)
NEXT I
PRINT
FOR I = N TO 1 STEP -1
  PRINT A$(I)
NEXT I
```

```
'1.9
' This program displays a large X made up of letter X's.
'
INPUT "Enter number of X's:"; NUM
CLS
FOR I = 1 TO NUM
  LOCATE I, I: PRINT "X"
  LOCATE I, NUM - I + 1: PRINT "X"
NEXT I
```

```
'1.10
' This program will display the savings in postage.
'
COST = 23.33333
INPUT "Enter # of printed sides:"; PS
INPUT "Enter # of single sided pages:"; SS
' Calculate # of pages and weight for 1st bill
PAGE1 = PS - 6: OZ1 = 1
OZ1 = OZ1 + INT((PAGE1 + 8) / 9)
' Calculate # of pages and weight for 2nd bill
PAGE2 = SS + INT((PS - SS + 1) / 2) - 6
OZ2 = 1
OZ2 = OZ2 + INT((PAGE2 + 8) / 9)
PRINT USING "###.## CENTS SAVED"; (OZ1 - OZ2) * COST
```

```
'2.1
' This program finds integral solutions of (X,Y) for AX + BY = C.
'
INPUT "Enter A, B, C:"; A, B, C
X = 1
DO
  Y = (C - A * X) / B
  IF ABS(Y - INT(Y)) < .001 THEN
    PRINT "("; LTRIM$(STR$(X)); ","; LTRIM$(STR$(Y)); ")"
    END
  END IF
  X = X + 1
LOOP UNTIL X > 10000
```

```
'2.2
' This program verifies a part number by validating check digit
'
INPUT "Enter part number:"; PART$
L = LEN(PART$): PROD = 1
FOR I = 1 TO L - 1
  DIGIT = VAL(MID$(PART$, I, 1))
  SUM = SUM + DIGIT * ((I MOD 2) + 1)
NEXT I
' Subtract units digit of sum from 9 for check digit
CHKDIGIT = 9 - (SUM MOD 10)
IF CHKDIGIT = VAL(RIGHT$(PART$, 1)) THEN
  PRINT "OKAY"
ELSE
  PRINT "ERROR - CHECK DIGIT SHOULD BE"; CHKDIGIT
END IF
```

```
'2.3
' This program determines number of prizes given of $13 million
'
PRIZE = 13000000
' Same algorithm is used as converting # to base 13 #
FOR I = 6 TO 0 STEP -1
  POW(I) = INT(13 ^ I + .1)
  A(I) = INT(PRIZE / POW(I))
  PRIZE = PRIZE MOD POW(I)
NEXT I
FOR I = 0 TO 6
  PRINT "$"; LTRIM$(STR$(POW(I))); " ="; A(I)
NEXT I
```

```
'2.4
' This program will determine the cost of Directory Assistance.
'
INPUT "Enter number of DACs:"; N
FOR I = 1 TO N
  INPUT "Enter DAC:"; DAC$
  IF DAC$ = "00" THEN
    COST = 3!
  ELSEIF DAC$ = "1411" THEN
    LOCALDAC = LOCALDAC + 1: COST = 0
  ELSE
    AREA$ = MID$(DAC$, 2, 3)
    IF AREA$ = "813" THEN
      COST = .25
    ELSEIF AREA$ = "305" OR AREA$ = "407" OR AREA$ = "904" THEN
      COST = .4
    ELSE
      COST = .65
    END IF
  END IF
  TOT = TOT + COST
NEXT I
' Every local DAC after the third cost 25 cents
IF LOCALDAC > 3 THEN
  TOT = TOT + (LOCALDAC - 3) * .25
END IF
PRINT USING "##.## DOLLARS"; TOT
```

```
'2.5
' This program will display the heading of even/odd pages.
'
DATA PROBLEMS,180,JUDGING CRITERIA,140
DATA BASIC SOLUTIONS,200,PASCAL SOLUTIONS,260
FOR I = 1 TO 4: READ P$(I), PNUM(I): NEXT I
'
INPUT "Enter page number:"; PAGE
IF PAGE MOD 2 = 0 THEN
  PRINT LTRIM$(STR$(PAGE));
  PRINT "  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION";
  PRINT " 1985 - 1994"
ELSE
  PRINT "FHSCC '";
  I = 1: PAG = PAGE
  WHILE PAG > PNUM(I)
    PAG = PAG - PNUM(I): I = I + 1
  WEND
  CH = INT(PAG / (PNUM(I) / 10))
  PRINT USING "## "; 85 + CH;
  PRINT P$(I); " ";
  PRINT PAGE
END IF
```

```
'2.6
' This program will compute the total ESTIMATED PREPARATION TIME.
'
DATA 1040,A,B,C,D,E
DATA 3,8,  2,53, 4,41, 0,53
DATA 2,32, 0,26, 1,10, 0,27
DATA 0,33, 0,8,  0,17, 0,20
DATA 6,26, 1,10, 2,5,  0,35
DATA 0,51, 0,42, 1,1,  0,41
DATA 2,52, 1,7, 1,16,  0,35
FOR I = 1 TO 6: READ FORM$(I): NEXT I
FOR I = 1 TO 6
  FOR J = 1 TO 4
    READ HR(I, J), MIN(I, J)
  NEXT J
NEXT I
' Tally form times until invalid entry
I = 0
DO UNTIL I > 6
  INPUT "Enter form:"; F$
  I = 1
  WHILE (I < 7) AND (F$ <> FORM$(I)): I = I + 1: WEND
  IF I < 7 THEN
    FOR J = 1 TO 4
      TOTHR = TOTHR + HR(I, J)
      TOTMIN = TOTMIN + MIN(I, J)
    NEXT J
  END IF
LOOP
'
TOTHR = TOTHR + INT(TOTMIN / 60)
TOTMIN = TOTMIN MOD 60
PRINT TOTHR; "HR.,"; TOTMIN; "MIN."
```

```
'2.7
' This program will calculate investments at GTE.
'
BEGPRICE = 32! * .85
RETURN401K = .14
'
INPUT "Enter salary:"; SALARY
INPUT "Enter 401K %:"; PERCENT: PERCENT = PERCENT / 100
MAXSHARES = INT(SALARY / 100)
PRINT "YOU CAN PURCHASE UP TO"; MAXSHARES; "SHARES"
INPUT "Enter number of shares:"; SHARES
INPUT "Enter end of year price:"; ENDPRICE
'
EMPCONT = SALARY * PERCENT
IF PERCENT >= .06 THEN
  COMPCONT = (SALARY * .06) * .75
ELSE
  COMPCONT = (SALARY * PERCENT) * .75
END IF
K401 = (EMPCONT + COMPCONT) * RETURN401K
STOCKGAIN = SHARES * (ENDPRICE - BEGPRICE)
TOTALGAIN = COMPCONT + K401 + STOCKGAIN
'
PRINT USING "COMPANY CONTRIBUTION: #####.##"; COMPCONT
PRINT USING "401K INTEREST RETURN: #####.##"; K401
PRINT USING "         STOCK GAINS: #####.##"; STOCKGAIN
PRINT USING "         TOTAL GAINS: #####.##"; TOTALGAIN
```

```
'2.8
' This program will produce loops of a spiral using letters.
'
INPUT "Enter number of spiral loops:"; NUM
INPUT "Enter first letter:"; LET$
CLS
ROW = 12: COL = 40: INCR = 1
WHILE LOOPNUM < NUM
  INCR = INCR + 2
  ' Go right
  LOCATE ROW, COL: PRINT STRING$(INCR, LET$)
  COL = COL + INCR - 1
  ' Go down
  FOR I = 1 TO INCR - 1
    LOCATE ROW + I, COL: PRINT LET$
  NEXT I
  ROW = ROW + INCR - 1: INCR = INCR + 2
  ' Go left
  COL = COL - INCR + 1
  LOCATE ROW, COL: PRINT STRING$(INCR, LET$)
  ' Go up
  FOR I = 1 TO INCR - 2
    LOCATE ROW - I, COL: PRINT LET$
  NEXT I
  ROW = ROW - INCR + 1
  IF LET$ = "Z" THEN LET$ = "A" ELSE LET$ = CHR$(ASC(LET$) + 1)
  LOOPNUM = LOOPNUM + 1
WEND
```

```
'2.9
' This program shows all possible moves for a Queen in chess.
'
INPUT "Enter column and row:"; RC$
COL = ASC(LEFT$(RC$, 1)) - ASC("A") + 1
ROW = 9 - VAL(RIGHT$(RC$, 1))
CLS
FOR I = 8 TO 1 STEP -1: PRINT USING "#"; I: NEXT I
PRINT "  A B C D E F G H"
' Horizontal moves
LOCATE ROW, 3: PRINT "* * * * * * * *"
' Vertical moves
FOR I = 1 TO 8: LOCATE I, COL * 2 + 1: PRINT "*": NEXT I
' Diagonal moves
FOR I = 1 TO 7
  R(1) = ROW - I: C(1) = COL - I
  R(2) = ROW + I: C(2) = COL + I
  R(3) = ROW - I: C(3) = COL + I
  R(4) = ROW + I: C(4) = COL - I
  FOR J = 1 TO 4
    IF R(J) > 0 AND R(J) < 9 AND C(J) > 0 AND C(J) < 9 THEN
       LOCATE R(J), C(J) * 2 + 1: PRINT "*"
    END IF
  NEXT J
NEXT I
LOCATE ROW, COL * 2 + 1: PRINT "Q"
```

```
'2.10
' This program tabulates information during a pre-election.
'
DATA MALE,FEMALE,50 AND BELOW,OVER 50,WHITE,OTHERS
DATA ABOVE $25000,$25000 AND BELOW
DATA WHITE MALE OVER 50 AND ABOVE $25000,OTHER
INPUT "Enter sex:"; SEX$
WHILE SEX$ <> "E"
  INPUT "Enter age:"; AGE
  INPUT "Enter race:"; RACE$
  INPUT "Enter income:"; INCOME
  INPUT "Enter party:"; PARTY$
  IF PARTY$ = "D" THEN COL = 1 ELSE COL = 2
  IF SEX$ = "M" THEN ROW = 1 ELSE ROW = 2
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF AGE <= 50 THEN ROW = 3 ELSE ROW = 4
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF RACE$ = "W" THEN ROW = 5 ELSE ROW = 6
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF INCOME > 25000 THEN ROW = 7 ELSE ROW = 8
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  IF RACE$ = "W" AND SEX$ = "M" AND AGE > 50 AND ROW = 7 THEN
    ROW = 9
  ELSE
    ROW = 10
  END IF
  SUM(ROW, COL) = SUM(ROW, COL) + 1
  TOTAL = TOTAL + 1: PRINT
  INPUT "Enter sex:"; SEX$
WEND
'
PRINT TAB(33); "DEMOCRATIC  REPUBLICAN";
FOR ROW = 1 TO 10
  IF ROW MOD 2 = 1 THEN PRINT
  READ A$(ROW): PRINT A$(ROW);
  PRINT TAB(38);
  PRINT USING "###.#"; SUM(ROW, 1) / TOTAL * 100;
  PRINT USING "       ###.#"; SUM(ROW, 2) / TOTAL * 100
NEXT ROW
```

```
'3.1
' This program will determine how much IRS owes/pays.
'
DATA 22750, 55100, 115000, 250000, 9999999
FOR I = 1 TO 5: READ AMOUNT(I): NEXT I
DATA .15, .28, .31, .36, .396
FOR I = 1 TO 5: READ RATE(I): NEXT I
STDEDUCT = 3800: EXEMPTION = 2450
'
INPUT "Enter adjusted gross income:"; GROSS
INPUT "Enter itemized deductions:"; DEDUCTIONS
INPUT "Enter federal income tax withheld"; FEDTAX
IF DEDUCTIONS > STDEDUCT THEN
  INCOME = GROSS - DEDUCTIONS
ELSE
  INCOME = GROSS - STDEDUCT
END IF
TAXINC = INCOME - EXEMPTION
'
FOR I = 1 TO 5
  IF TAXINC <= AMOUNT(I) THEN
    FOR J = 1 TO I - 1
      TAX = TAX + (AMOUNT(J) - AMOUNT(J - 1)) * RATE(J)
    NEXT J
    TAX = TAX + (TAXINC - AMOUNT(I - 1)) * RATE(I)
    PRINT USING "######.## DOLLARS "; ABS(TAX - FEDTAX);
    IF FEDTAX < TAX THEN
      PRINT "YOU OWE"
    ELSE
      PRINT "WILL BE REFUNDED TO YOU"
    END IF: END
  END IF
NEXT I
```

```
'3.2
' This program will display a simplified phone bill.
'
L = 1: INPUT "Enter MIN:"; MIN(L)
WHILE MIN(L) > 0
  INPUT "Enter time:"; TIM$(L)
  L = L + 1
  INPUT "Enter MIN:"; MIN(L)
WEND
L = L - 1
'
' Display bill
PRINT "  BOB SMITH  (813) 555-1234": PRINT
PRINT "  TIME OF DAY  MIN.  CHARGE"
FOR I = 1 TO L
  IF LEFT$(TIM$(I), 1) = "0" THEN
    PRINT " "; MID$(TIM$(I), 2);
  ELSE
    PRINT TIM$(I);
  END IF
  ' Calculate charge
  HH = VAL(LEFT$(TIM$(I), 2))
  AM$ = MID$(TIM$(I), 7, 2)
  DAY$ = RIGHT$(TIM$(I), 3)
  BOOL1 = (HH > 7 AND HH < 12 AND AM$ = "AM")
  BOOL2 = (HH = 12 AND AM$ = "PM") OR (HH < 5 AND AM$ = "PM")
  MIDDAY = BOOL1 OR BOOL2
  IF HH > 4 AND HH < 11 AND AM$ = "PM" AND DAY$ <> "SAT" THEN
    RATE1 = .21: RATE2 = .16
  ELSEIF MIDDAY AND DAY$ <> "SAT" AND DAY$ <> "SUN" THEN
    RATE1 = .28: RATE2 = .21
  ELSE
    RATE1 = .14: RATE2 = .11
  END IF
  CHARGE(I) = RATE1 + RATE2 * (MIN(I) - 1)
  PRINT USING "  ###"; MIN(I);
  PRINT USING "   ###.##"; CHARGE(I)
  TOT = TOT + CHARGE(I)
NEXT I
IF TOT > 20 THEN DISC = TOT * .2
PRINT
PRINT "TOTAL CHARGES"; TAB(22);
PRINT USING "###.##"; TOT
PRINT "DISCOUNT"; TAB(22);
PRINT USING "###.##"; DISC
PRINT "CHARGES - DISCOUNT"; TAB(22);
PRINT USING "###.##"; TOT - DISC
```

```
'3.3
' This program simulates a baseball game.
'
DEFINT A-W
RANDOMIZE TIMER
CLS : PRINT
PRINT SPACE$(8);
FOR I = 1 TO 9: PRINT I; : NEXT I: PRINT " SCORE"
PRINT SPACE$(8); : FOR I = 1 TO 33: PRINT "-"; : NEXT I: PRINT
PRINT "TEAM A !"; SPACE$(27); "!"
PRINT "TEAM B !"; SPACE$(27); "!"
FOR IN = 1 TO 9
  FOR T = 1 TO 2
    S = 0: B = 0: W = 0: R = 0: O = 0
    WHILE O < 3
      X = RND(3)
      IF X < .4 THEN S = S + 1: STOT = STOT + 1
      IF X >= .4 THEN B = B + 1: BTOT = BTOT + 1
      IF S = 3 THEN O = O + 1: OTOT = OTOT + 1: S = 0: W = 0
      IF B = 4 THEN W = W + 1: WTOT = WTOT + 1: B = 0: S = 0
      IF W = 4 THEN R = R + 1: R(T) = R(T) + 1: W = 3
    WEND
    LOCATE 3 + T, 6 + IN * 3: PRINT R;
  NEXT T
NEXT IN
LOCATE 4, 39: PRINT USING "##"; R(1)
LOCATE 5, 39: PRINT USING "##"; R(2)
PRINT
PRINT "TOTAL # OF STRIKES:"; STOT
PRINT "TOTAL # OF BALLS:"; BTOT
PRINT "TOTAL # OF WALKS:"; WTOT
PRINT "TOTAL # OF STRIKE OUTS:"; OTOT
```

```
'3.4
' This program will produce all possible subsets of letters.
'
DEFINT A-Z: DIM SUB$(1024)
INPUT "Enter letters:"; L$
L = LEN(L$)
FOR I = 1 TO L: A$(I) = MID$(L$, I, 1): NEXT I
'
' Sort letters in A$()
FOR I = 1 TO L - 1
  FOR J = I + 1 TO L
    IF A$(I) > A$(J) THEN SWAP A$(I), A$(J)
  NEXT J
NEXT I
'
' Generate binary numbers to produce all subsets.
FOR N = 0 TO 2 ^ L - 1
  NUM = N
  FOR J = L - 1 TO 0 STEP -1
    BIT = INT(NUM / 2 ^ J)
    IF BIT THEN
      SUB$(N) = SUB$(N) + A$(L - J): NUM = NUM - 2 ^ J
    END IF
  NEXT J
NEXT N
'
' Bubble Sort subsets
FOR I = 0 TO 2 ^ L - 2
  FOR J = I + 1 TO 2 ^ L - 1
    IF SUB$(I) > SUB$(J) THEN SWAP SUB$(I), SUB$(J)
  NEXT J
NEXT I
'
' Display subsets
FOR I = 0 TO 2 ^ L - 1
  SUBLEN = LEN(SUB$(I)) + 3
  IF COL + SUBLEN > 50 THEN PRINT : COL = 0
  PRINT "{"; SUB$(I); "} ";
  COL = COL + SUBLEN
NEXT I
PRINT : PRINT "TOTAL SUBSETS ="; 2 ^ L
```

```
'3.5
' This program will sum big integers from 1 to N.
' Gauss's formula: SUM = N * (N+1) / 2.
'
DIM A(80), B(80), PROD(80), D(80)
INPUT "Enter N:"; N$
'
' Store digits of N$ in A() and B()
LENA = LEN(N$): LENB = LENA
FOR I = 1 TO LENA
  A(I) = VAL(MID$(N$, LENA - I + 1, 1))
  B(I) = A(I)
NEXT I
'
' Add 1 to number in B()
B(1) = B(1) + 1: I = 1
WHILE B(I) = 10
  B(I) = 0: I = I + 1: B(I) = B(I) + 1
WEND
IF I > LENB THEN LENB = I
'
' Multiply A() by B()
FOR I = 1 TO LENA
  CARRY = 0
  FOR J = 1 TO LENB
    S = I + J - 1
    PROD(S) = PROD(S) + A(I) * B(J) + CARRY
    CARRY = INT(PROD(S) / 10)
    PROD(S) = PROD(S) - CARRY * 10
  NEXT J
  IF CARRY > 0 THEN PROD(S + 1) = CARRY
NEXT I
IF CARRY > 0 THEN S = S + 1
'
' Divide product PROD() by 2
IF PROD(S) = 1 THEN S = S - 1: CARRY = 10
FOR I = S TO 1 STEP -1
  D(I) = INT((PROD(I) + CARRY) / 2)
  CARRY = (PROD(I) MOD 2) * 10
NEXT I
'
' Display answer in D()
FOR I = S TO 1 STEP -1
  PRINT USING "#"; D(I);
NEXT I: PRINT
```

```
'3.6
' This program will assign values to variables in BASIC code.
'
DO
  L = L + 1
  INPUT "Enter line:"; A$(L)
LOOP UNTIL A$(L) = "END"
L = L - 1
'
FOR I = 1 TO L
  ' Determine if first variable is new or old
  V$ = LEFT$(A$(I), 1)
  POSV = INSTR(ALLV$, V$)
  IF POSV = 0 THEN
    ALLV$ = ALLV$ + V$
    POSV = LEN(ALLV$)
  END IF
  '
  ' Assign value for first number
  CH$ = MID$(A$(I), 3, 1)
  IF CH$ >= "0" AND CH$ <= "9" THEN
    NUM1 = VAL(CH$)
  ELSE
    POSV2 = INSTR(ALLV$, CH$)
    NUM1 = B(POSV2)
  END IF
  '
  IF LEN(A$(I)) = 3 THEN
    ' Assign first number to current variable
    B(POSV) = NUM1
  ELSE
    ' Assign value for second number
    CH$ = RIGHT$(A$(I), 1)
    IF CH$ >= "0" AND CH$ <= "9" THEN
      NUM2 = VAL(CH$)
    ELSE
      POSV3 = INSTR(ALLV$, CH$)
      NUM2 = B(POSV3)
    END IF
    ' Perform operation with 1st and 2nd num and place in var
    OP$ = MID$(A$(I), 4, 1)
    SELECT CASE OP$
      CASE "+": B(POSV) = NUM1 + NUM2
      CASE "-": B(POSV) = NUM1 - NUM2
      CASE "*": B(POSV) = NUM1 * NUM2
      CASE "/": B(POSV) = NUM1 / NUM2
    END SELECT
  END IF
NEXT I
' Display the variables in order of appearance with values
FOR I = 1 TO LEN(ALLV$)
  PRINT MID$(ALLV$, I, 1); "=";
  PRINT LTRIM$(STR$(B(I)))
NEXT I
```

```
'3.7
' This program finds three 3-digit primes having digits 1-9.
'
' Generate primes into A()
DEFINT B-Z: DEFLNG A: DIM A(200)
FOR I = 101 TO 997 STEP 2
  J = 3: PRIME = -1
  WHILE (J <= SQR(I)) AND PRIME
    IF I MOD J = 0 THEN PRIME = 0
    J = J + 2
  WEND
  IF PRIME THEN  'Ensure that Digits are unique and not 0
    H = INT(I / 100)
    T = INT((I - H * 100) / 10)
    ONE = I - H * 100 - T * 10
    IF T > 0 AND H <> T AND T <> ONE AND H <> ONE THEN
      P = P + 1: A(P) = I
    END IF
  END IF
NEXT I
' Add the different combinations of 3 primes
FOR I = 1 TO P - 2
  FOR J = I + 1 TO P - 1
    FOR K = J + 1 TO P
      SUM = A(I) + A(J) + A(K)
      ' Check if SUM has 4 digits in ascending order
      IF SUM >= 1234 THEN
        DIGITS$ = LTRIM$(STR$(SUM)): GOOD = -1: L = 1
        DO
          IF MID$(DIGITS$, L, 1) >= MID$(DIGITS$, L + 1, 1) THEN
            GOOD = 0
          END IF
          L = L + 1
        LOOP UNTIL (L = 4) OR NOT GOOD
        ' Check all 3-digit primes for digits 1 through 9
        IF GOOD THEN
          ADIGITS = (A(I) * 1000 + A(J)) * 1000 + A(K)
          DIGITS$ = LTRIM$(STR$((ADIGITS))): L = 1
          WHILE (L <= 9) AND GOOD
            IF INSTR(DIGITS$, CHR$(48 + L)) = 0 THEN GOOD = 0
            L = L + 1
          WEND
          IF GOOD THEN
            PRINT A(I); "+"; A(J); "+"; A(K); "="; SUM
            PNUM = PNUM + 1: IF PNUM = 7 THEN END
          END IF
        END IF
      END IF
    NEXT K
  NEXT J
NEXT I
```

```
'3.8
' This program will display time MM:SS in block letters.
'
DATA ****     *  ****  ****  *  *  ****  *     ****  ****  ****
DATA *  *     *     *     *  *  *  *     *        *  *  *  *  *
DATA *  *     *  ****  ****  ****  ****  ****     *  ****  ****
DATA *  *     *  *        *     *     *  *  *     *  *  *     *
DATA ****     *  ****  ****     *  ****  ****     *  ****     *
DATA 6,10,6,10, 1,7,18,24
FOR I = 1 TO 5
  READ B$
  FOR J = 0 TO 9: A$(I, J) = MID$(B$, J * 6 + 1, 4): NEXT J
NEXT I
FOR I = 1 TO 4: READ MAX(I): NEXT I  'Maximum units for MM:SS
FOR I = 1 TO 4: READ COL(I): NEXT I  'Columns to start blocks
'
INPUT "Enter MM:SS:"; MMSS$
FOR I = 1 TO 4
  DIG(I) = VAL(MID$(MMSS$, I - (I > 2), 1))
NEXT I
'
CLS
LOCATE 2, 14: PRINT "*": LOCATE 4, 14: PRINT "*"
DO UNTIL CH$ <> ""
  FOR I = 1 TO 4
    FOR J = 1 TO 5
      LOCATE J, COL(I): PRINT A$(J, DIG(I))
    NEXT J
  NEXT I
  DIG(4) = DIG(4) + 1
  FOR J = 4 TO 1 STEP -1
    IF DIG(J) = MAX(J) THEN
      DIG(J - 1) = DIG(J - 1) + 1: DIG(J) = 0
    END IF
  NEXT J
  FOR I = 1 TO 3000: NEXT I  'Approximately 1 second
  CH$ = INKEY$
LOOP
```

```
'3.9
' This program will calculate the area of a polygon room.
'
INPUT "Enter number of sides:"; SIDES
FOR I = 1 TO SIDES
  INPUT "Enter movement:"; MOV$
  DIR$(I) = MID$(MOV$, 1, 1)
  L = LEN(MOV$)
  MOV$ = MID$(MOV$, 2, L - 1)
  DIST(I) = VAL(MOV$)
' Subtract Down and Left directions
  IF DIR$(I) = "D" OR DIR$(I) = "L" THEN DIST(I) = -DIST(I)
NEXT I
' Multiply length by width to obtain rectangle area,
' then add or subtract area from overall area.
I = 1: SUM = 0: AREA = 0
WHILE (I <= SIDES)
  SUM = SUM + DIST(I)
  AREA = AREA + (SUM * DIST(I + 1))
  I = I + 2
WEND
PRINT "AREA ="; ABS(AREA); "SQUARE FEET"
```

```
'3.10
' This program displays versions of libraries on a graph.
'
INPUT "Enter version #:"; Vers
INPUT "Enter first week in test:"; FirstWk
INPUT "Enter first week to display, # of weeks:"; FWKDisp, WkNum
CLS
LWKDisp = FWKDisp + WkNum - 1
' Display week #s at top (units first, then tens)
PRINT SPACE$(9);
FOR I = FWKDisp TO LWKDisp
  PRINT USING "#"; INT(I / 10);
NEXT I
PRINT : PRINT SPACE$(9);
FOR I = FWKDisp TO LWKDisp
  PRINT USING "#"; I MOD 10;
NEXT I
PRINT : PRINT
LastWk = FirstWk + 17
' Compute # of versions to backup from Vers input
Backup = INT((LastWk - FWKDisp) / 6)
Vers = Vers - Backup
FirstWk = FirstWk - 6 * Backup: LastWk = LastWk - 6 * Backup
DO UNTIL FirstWk > LWKDisp
  ' Display Version and indent
  PRINT "R1V"; RIGHT$(STR$(100 + Vers), 2); "L01 ";
  IF FWKDisp <= FirstWk THEN
    Min = FirstWk
    PRINT SPACE$(FirstWk - FWKDisp);
  ELSE
```

```
    Min = FWKDisp
  END IF
  IF LWKDisp >= LastWk THEN Max = LastWk ELSE Max = LWKDisp
  ' Display TestArea of 1 if Vers even, 2 if odd; P = Production
  TestArea = (Vers MOD 2) + 1
  FOR I = Min TO Max
    IF I < FirstWk + 12 THEN
      PRINT USING "#"; TestArea;
    ELSE
      PRINT "P";
    END IF
  NEXT I
  PRINT
  ' Display Pre-Production Version
  FirstPreWk = FirstWk + 5: LastPreWk = FirstWk + 10
  IF (LastPreWk >= FWKDisp) AND (FirstPreWk <= LWKDisp) THEN
    PRINT "R1V"; RIGHT$(STR$(100 + Vers - 1), 2); "L88 ";
    IF FirstPreWk > FWKDisp THEN
      Min = FirstPreWk
      PRINT SPACE$(FirstPreWk - FWKDisp);
    ELSE
      Min = FWKDisp
    END IF
    IF LWKDisp >= LastPreWk THEN
      Max = LastPreWk
    ELSE
      Max = LWKDisp
    END IF
    PRINT STRING$(Max - Min + 1, "*")
  END IF
  FirstWk = FirstWk + 6: LastWk = LastWk + 6
  Vers = Vers + 1
LOOP
```

FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '96
BASIC PROGRAM SOLUTIONS

```
'1.1
' This program displays a phrase of the form FHSCC '##.
'
INPUT "Enter year:"; YEAR$
PRINT "FHSCC '"; MID$(YEAR$, 3, 2)
```

```
'1.2
' This program tallies number of frequent flier miles.
'
INPUT "Enter X:"; X
INPUT "Enter Y:"; Y
PRINT X * (1300 + 1300 + 500) + (Y * 5)
```

```
'1.3
' This program displays middle letter(s) of a word.
'
INPUT "Enter word:"; WORD$
L = LEN(WORD$): M = INT(L / 2)
IF L MOD 2 = 0 THEN PRINT MID$(WORD$, M, 1);
PRINT MID$(WORD$, M + 1, 1)
```

```
'1.4
' This program displays area and perimeter of a rectangle
'
INPUT "Enter coordinate 1:"; X1, Y1
INPUT "Enter coordinate 2:"; X2, Y2
AREA = ABS((X1 - X2) * (Y1 - Y2))
PERIM = (ABS(X1 - X2) + ABS(Y1 - Y2)) * 2
PRINT "AREA ="; AREA
PRINT "PERIMETER ="; PERIM
```

```
'1.5
' This program code-breaks an encrypted secret message.
'
INPUT "Enter encryption:"; E$
FOR I = 1 TO LEN(E$)
  M$ = MID$(E$, I, 1)
  IF M$ = " " THEN
    PRINT M$;
  ELSE
    PRINT CHR$(ASC("Z") - ASC(M$) + ASC("A"));
  END IF
NEXT I
PRINT
```

```
'1.6
' This program display number of floors touched by elevator
'
DO
  INPUT "Enter floor:"; FLOOR
  TOTAL = TOTAL + ABS(FLOOR - LASTFLOOR)
  IF FLOOR > MAX THEN MAX = FLOOR
  LASTFLOOR = FLOOR
LOOP UNTIL (FLOOR = 0)
' 1 is added for the starting ground floor
PRINT "TOTAL FLOORS TOUCHED ="; TOTAL + 1
PRINT "UNIQUE FLOORS TOUCHED ="; MAX + 1
```

```
'1.7
' This program displays a person's ratios for buying a house.
'
INPUT "Enter amount of loan:"; LOAN
INPUT "Enter amount of debts:"; DEBTS
INPUT "Enter amount of income:"; INCOME
RATIO1 = (LOAN / INCOME) * 100
RATIO2 = ((LOAN + DEBTS) / INCOME) * 100
PRINT USING "RATIOS = ##.#% / ##.#%"; RATIO1; RATIO2
PRINT "DOES ";
IF RATIO1 > 33 OR RATIO2 > 38 THEN PRINT "NOT ";
PRINT "QUALIFY"
```

```
'1.8
' This program will convert numbers to English or Spanish.
'
DATA ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE,TEN
DATA UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE,DIEZ
INPUT "Enter E or S:"; LANG$
INPUT "Enter number:"; NUM
IF LANG$ = "S" THEN FOR I = 1 TO 10: READ N$: NEXT I
FOR I = 1 TO NUM
  READ N$
NEXT I
PRINT N$
```

```
'1.9
' This program forms a cross from word(s).
'
INPUT "Enter word(s):"; W$
L = LEN(W$): M = INT(L / 2) + 1
FOR I = 1 TO L
  IF I <> M THEN
    PRINT SPACE$(M - 1); MID$(W$, I, 1)
  ELSE
    PRINT W$
  END IF
NEXT I
```

```
'1.10
' This program simulates the PRICE IS RIGHT game.
'
INPUT "Enter actual price:"; PRICE
INPUT "Enter guesses A, B, C, D"; A(1), A(2), A(3), A(4)
MIN = 32000
FOR I = 1 TO 4
  IF A(I) <= PRICE THEN
    DIF = PRICE - A(I)
    IF DIF < MIN THEN MIN = DIF: INDEX = I
  END IF
NEXT I
IF INDEX > 0 THEN
  PRINT "PERSON "; MID$("ABCD", INDEX, 1)
ELSE
  PRINT "EVERYONE IS OVER"
END IF
```

```
'2.1
' This program will emulate random dart throws.
'
DATA 0,2,4,5,10,20,50
FOR I = 1 TO 7: READ S$(I): NEXT I: PRINT " ";
RANDOMIZE TIMER
DO
  X = INT(RND(3) * 7) + 1: THROW = THROW + 1
  PRINT S$(X);
  TOTAL = TOTAL + VAL(S$(X))
  IF TOTAL < 100 THEN PRINT ",";
LOOP UNTIL TOTAL >= 100
PRINT : PRINT THROW; "THROWS ACHIEVED SCORE OF"; TOTAL: PRINT
```

```
'2.2
' This program compresses information to save space.
'
INPUT "Enter string:"; S$
FOR I = 1 TO LEN(S$)
  MD$ = MID$(S$, I, 1)
  IF MD$ <> "*" THEN
    IF AST > 0 THEN
      IF AST = 1 THEN PRINT "*";  ELSE PRINT USING "#"; AST;
      AST = 0
    END IF
    PRINT MD$;
  ELSE
    AST = AST + 1
  END IF
NEXT I
PRINT
```

```
'2.3
' This program finds 2 numbers to add to the set 1,3,8.
'
A(1) = 1: A(2) = 3: A(3) = 8: N = 3: I = 0
FOR I = 0 TO 999
  FOUND = -1
  FOR J = 1 TO N
    NUM = A(J) * I + 1
    IF SQR(NUM) - INT(SQR(NUM + .0001)) > .0001 THEN FOUND = 0
  NEXT J
  IF FOUND THEN
    PRINT I: N = N + 1: A(N) = I: IF N = 5 THEN END
  END IF
NEXT I
```

```
'2.4
' This program diplays the LCM of the first N integers.
'
DIM A(31): DEFDBL P
INPUT "Enter N:"; N
FOR I = 2 TO N: A(I) = I: NEXT I
' Produce all the necessary prime factors
FOR I = 2 TO N
  FOR J = I + 1 TO N
    IF A(J) MOD A(I) = 0 THEN A(J) = A(J) / A(I)
  NEXT J
NEXT I
'
PROD = 1
FOR I = 2 TO N: PROD = PROD * A(I): NEXT I
PRINT PROD
```

```
'2.5
' This program will calculate the fractional value.
'
INPUT "Enter word: "; A$
FOR I = 1 TO 3
  A(I) = ASC(MID$(A$, I, 1)) - 64
NEXT I
N = A(1) * A(2) + A(2) * A(3) + A(1) * A(3)
D = A(1) * A(2) * A(3)
FOR I = D TO 1 STEP -1
  IF N MOD I = 0 AND D MOD I = 0 THEN
    PRINT LTRIM$(STR$(N / I)); "/"; LTRIM$(STR$(D / I)): END
  END IF
NEXT I
```

```
'2.6
' This program displays the Nth prime in Fibonacci sequence.
'
DIM F(99)
F(1) = 1: F(2) = 1: F(3) = 2: PNUM = 1: I = 3
INPUT "Enter N:"; N
WHILE PNUM < N
  I = I + 1
  F(I) = F(I - 1) + F(I - 2): PRIME = -1
  ' Check if Fibonacci # is prime (not divisible by 2 or odd #)
  IF F(I) MOD 2 = 0 THEN PRIME = 0
  IF PRIME THEN
    FOR J = 3 TO SQR(F(I))
      IF F(I) MOD J = 0 THEN PRIME = 0
    NEXT J
    IF PRIME THEN PNUM = PNUM + 1
  END IF
WEND
PRINT F(I)
```

```
'2.7
' This program sorts phone bills by zip code and phone #.
'
DO
  N = N + 1
  INPUT "Enter phone #, zip:"; P$(N), Z$(N)
  PZ$(N) = Z$(N) + P$(N)
LOOP UNTIL (P$(N) = "0000") AND (Z$(N) = "00000")
N = N - 1
FOR I = 1 TO N - 1
  FOR J = I + 1 TO N
    IF PZ$(I) > PZ$(J) THEN
      SWAP PZ$(I), PZ$(J)
      SWAP P$(I), P$(J)
      SWAP Z$(I), Z$(J)
    END IF
  NEXT J
NEXT I
FOR I = 1 TO N: PRINT P$(I): NEXT I
```

```
'2.8
' This program will display number of runs of letters.
'
INPUT "Enter letters:"; LET$
FOR I = 1 TO LEN(LET$)
  CH$ = MID$(LET$, I, 1)
  IF INSTR("ABCDEFGHIJKLM", CH$) > 0 THEN
    IF HALF2 THEN H2 = H2 + 1: HALF2 = 0
    HALF1 = -1
  ELSE
    IF HALF1 THEN H1 = H1 + 1: HALF1 = 0
    HALF2 = -1
  END IF
NEXT I
IF HALF1 THEN H1 = H1 + 1
IF HALF2 THEN H2 = H2 + 1
PRINT "RUNS IN 1ST HALF ="; H1
PRINT "RUNS IN 2ND HALF ="; H2
```

```
'2.9
' This program reverses the order of letters in each word.
'
INPUT "Enter string:"; S$: S$ = S$ + " "
FOR I = 1 TO LEN(S$)
  MD$ = MID$(S$, I, 1)
  IF MD$ = " " THEN
    L = LEN(W$): PAL = -1
    FOR J = 1 TO L / 2
      IF MID$(W$, J, 1) <> MID$(W$, L - J + 1, 1) THEN PAL = 0
    NEXT J
    IF PAL THEN
      PRINT STRING$(LEN(W$), "?");
    ELSE
      FOR J = L TO 1 STEP -1: PRINT MID$(W$, J, 1); : NEXT J
    END IF
    PRINT " "; : W$ = ""
  ELSE
    W$ = W$ + MD$
  END IF
NEXT I
PRINT
```

```
'2.10
' This program determines day of week for a given date.
'
DIM MONNUM(12)
DATA 1,4,4,0,2,5,0,3,6,1,4,6
FOR I = 1 TO 12: READ MONNUM(I): NEXT I
INPUT "Enter month, day, year:"; MONTH, DAY, YEAR
LAST2 = YEAR MOD 100
SUM = LAST2 + INT(LAST2 / 4)
LEAPYEAR = (YEAR MOD 4 = 0) AND (YEAR MOD 100 > 0)
LEAPYEAR = LEAPYEAR OR (YEAR MOD 400 = 0)
IF (MONTH < 3) AND LEAPYEAR THEN
  IF MONTH = 2 THEN SUM = SUM + 3   'New Month Number
ELSE
  SUM = SUM + MONNUM(MONTH)
END IF
SUM = SUM + DAY
SELECT CASE YEAR
  CASE IS < 1800: SUM = SUM + 4
  CASE IS < 1900: SUM = SUM + 2
  CASE IS < 2000:
  CASE IS < 2100: SUM = SUM + 6
  CASE IS < 2200: SUM = SUM + 4
END SELECT
R = SUM MOD 7
DATA SATURDAY,SUNDAY,MONDAY,TUESDAY,WEDNESDAY,THURSDAY,FRIDAY
FOR I = 1 TO R + 1: READ D$: NEXT I
PRINT D$
```

```
'3.1
' This program displays the appearance of 3-dimensional book.
'
INPUT "Enter title 1:"; T1$
INPUT "Enter title 2:"; T2$
IF LEN(T1$) > LEN(T2$) THEN
  MAX = LEN(T1$): DIF = INT((MAX - LEN(T2$)) / 2)
  T2$ = SPACE$(DIF) + T2$ + SPACE$(DIF + 1)
ELSE
  MAX = LEN(T2$): DIF = INT((MAX - LEN(T1$)) / 2)
  T1$ = SPACE$(DIF) + T1$ + SPACE$(DIF + 1)
END IF
CLS
PRINT "    /---/!"
PRINT "   /   / !"
PRINT "  /   /  !"
PRINT " /   /   !"
PRINT "!---!    !"
FOR ROW = 1 TO MAX
  PRINT "!";
  PRINT MID$(T2$, ROW, 1); " ";
  PRINT MID$(T1$, ROW, 1); "!";
  IF ROW < MAX - 3 THEN
    PRINT SPACE$(4); "!"
  ELSE
    PRINT SPACE$(MAX - ROW + 1); "/"
  END IF
NEXT ROW
PRINT "!---!/"
```

```
'3.2
' This program produces a prime factors tree.
'
DIM P(100)
INPUT "Enter number:"; NUM
CLS : PRINT TAB(5); NUM
LEFT = 5: RIGHT = LEFT + LEN(STR$(NUM))  'Position of / and \
DO
  ' Find smallest prime that divides number
  IF NUM MOD 2 = 0 THEN
    PR = 2
  ELSE
    PR = 1
    DO
      PR = PR + 2
    LOOP UNTIL (NUM MOD PR = 0)
  END IF
  DIVIDEND = NUM / PR
  IF DIVIDEND > 1 THEN
    PRINT TAB(LEFT); "/"; TAB(RIGHT); "\"
    LNUM$ = LTRIM$(STR$(PR)): RNUM$ = LTRIM$(STR$(DIVIDEND))
    L = LEN(LNUM$): R = LEN(RNUM$)
    PRINT TAB(LEFT - L); LNUM$; TAB(RIGHT + 1); RNUM$
    LEFT = RIGHT: RIGHT = RIGHT + R + 1
  END IF
  NUM = DIVIDEND
LOOP UNTIL NUM = 1
```

```
'3.3
' This program simulates a "base four" calculator.
'
INPUT "Enter base 4 expression:"; E$: E$ = E$ + "+"
SYM$(1) = "+"
FOR I = 1 TO LEN(E$)
  CH$ = MID$(E$, I, 1)
  IF CH$ = "+" OR CH$ = "-" THEN
    J = J + 1: NUM$(J) = N$: SYM$(J + 1) = CH$: N$ = ""
  ELSE
    N$ = N$ + CH$
  END IF
NEXT I
' Convert base 4 numbers to base 10 and perform arithmetic
FOR I = 1 TO J
  L = LEN(NUM$(I)): B10 = 0
  FOR J = 1 TO L
    DIG = VAL(MID$(NUM$(I), J, 1))
    B10 = B10 + DIG * 4 ^ (L - J)
  NEXT J
  IF SYM$(I) = "-" THEN B10 = (-B10)
  TOTAL = TOTAL + B10
NEXT I
' Convert base 10 number to base 4
IF TOTAL < 0 THEN PRINT "-"; : TOTAL = (-TOTAL)
```

```
J = INT(LOG(TOTAL) / LOG(4) + .001)
FOR I = J TO 0 STEP -1
  POW = 4 ^ I
  X = INT(TOTAL / POW): PRINT USING "#"; X;
  TOTAL = TOTAL - X * POW
NEXT I
PRINT


'3.4
' This program calculates contractor's pay = time * rate
'
INPUT "Enter pay/hour:"; RATE
INPUT "Enter start time:"; ST$
INPUT "Enter finish time:"; FI$
STHOUR = VAL(MID$(ST$, 1, 2))
FIHOUR = VAL(MID$(FI$, 1, 2))
STMIN = VAL(MID$(ST$, 4, 2))
FIMIN = VAL(MID$(FI$, 4, 2))
' Adjust for 12AM and times from 1PM - 11PM
IF STHOUR = 12 THEN
  IF MID$(ST$, 6, 2) = "AM" THEN STHOUR = STHOUR - 12
ELSE
  IF MID$(ST$, 6, 2) = "PM" THEN STHOUR = STHOUR + 12
END IF
IF FIHOUR = 12 THEN
  IF MID$(FI$, 6, 2) = "AM" THEN FIHOUR = FIHOUR - 12
ELSE
  IF MID$(FI$, 6, 2) = "PM" THEN FIHOUR = FIHOUR + 12
END IF
' Adjust for a late starting time and early morning finish
IF STHOUR > FIHOUR THEN FIHOUR = FIHOUR + 24
' Compute difference in time (finish - start)
TIME = (FIHOUR - STHOUR) + (FIMIN - STMIN) / 60
' If more than half of time is outside normal hours (7AM - 5PM)
' then add a shift differential of 10% to rate.
IF (7 - STHOUR) + (0 - STMIN) / 60 >= TIME / 2 THEN
  ' More than half of time is worked before 7AM
  RATE = RATE * 1.1
END IF
IF (FIHOUR - 17) + (FIMIN) / 60 >= TIME / 2 THEN
  ' More than half of time is worked after 5PM
  RATE = RATE * 1.1
END IF
PRINT USING "$###.##"; TIME * RATE
```

```
'3.5
' This program will display the button that leads to the others.
'
FOR I = 1 TO 4
  INPUT "Enter row:"; ROW$
  FOR J = 1 TO 4
    N(I, J) = VAL(MID$(ROW$, J * 3 - 2, 1))
    D$(I, J) = MID$(ROW$, J * 3 - 1, 1)
  NEXT J
NEXT I
FOR I = 1 TO 4
 FOR J = 1 TO 4
   FOR K = 1 TO 4: FOR L = 1 TO 4: A(K, L) = 0: NEXT L, K
   R = I: C = J: A(R, C) = -1: PRESS = 1: GOOD = -1
   DO
     SELECT CASE D$(R, C)
       CASE "D": R = R + N(R, C)
       CASE "U": R = R - N(R, C)
       CASE "L": C = C - N(R, C)
       CASE "R": C = C + N(R, C)
     END SELECT
     IF A(R, C) THEN
       GOOD = 0
     ELSE
       A(R, C) = -1: PRESS = PRESS + 1
     END IF
   LOOP UNTIL (NOT GOOD) OR (PRESS = 16)
   IF PRESS = 16 THEN
     PRINT USING "FIRST BUTTON = #"; N(I, J); : PRINT D$(I, J)
     PRINT "AT ROW = "; : PRINT USING "#"; I;
     PRINT USING ", COL = #"; J: END
   END IF
 NEXT J
NEXT I
```

```
'3.6
' This program will generate odd size magic squares.
'
INPUT "Enter order, first number, increment: "; N, FIRST, INC
DIM A(N, N)
X = 1: Y = (N + 1) / 2: A(X, Y) = FIRST
FOR I = 2 TO N * N
  X = X - 1: Y = Y + 1
  IF X = 0 THEN X = N
  IF Y > N THEN Y = 1
  IF A(X, Y) = 0 THEN
    A(X, Y) = FIRST + INC * (I - 1)
  ELSE
    X = X + 2: Y = Y - 1
    IF X > N THEN X = X - N
    IF Y = 0 THEN Y = N
    A(X, Y) = FIRST + INC * (I - 1)
  END IF
NEXT I
' Display Magic Number and Square
FOR I = 1 TO N: MAGICNUM = MAGICNUM + A(I, 1): NEXT I
PRINT "MAGIC NUMBER ="; MAGICNUM
FOR I = 1 TO N
  FOR J = 1 TO N
    PRINT USING "####"; A(I, J);
  NEXT J: PRINT
NEXT I
```

```
'3.7
' This program will generate 6x6 magic squares.
'
INPUT "Enter first number, increment: "; FIRSTN, INC
' Four 3x3 squares are made for the 6x6 matrix B()
' upper-left, bottom-right, upper-right, bottom-left
DATA 0,0, 1,1, 0,1, 1,0
FOR SQ = 0 TO 3
  FIRST = FIRSTN + SQ * 9 * INC
  GOSUB Generate3x3
  READ R, C
  FOR I = 1 TO 3
    FOR J = 1 TO 3
      B(R * 3 + I, C * 3 + J) = A(I, J)
    NEXT J
  NEXT I
NEXT SQ
' Transpose three cells
SWAP B(1, 1), B(4, 1)
SWAP B(2, 2), B(5, 2)
SWAP B(3, 1), B(6, 1)
' Display 6x6 matrix
FOR I = 1 TO 6: MAGICNUM = MAGICNUM + B(I, 1): NEXT I
PRINT "MAGIC NUMBER ="; MAGICNUM
FOR I = 1 TO 6
  FOR J = 1 TO 6
    PRINT USING "####"; B(I, J);
  NEXT J: PRINT
NEXT I
END

Generate3x3:  'Generate a 3x3 magic square in A(1..3,1..3)
  FOR I = 1 TO 3: FOR J = 1 TO 3: A(I, J) = 0: NEXT J, I
  N = 3
  X = 1: Y = (N + 1) / 2: A(X, Y) = FIRST
  FOR I = 2 TO N * N
    X = X - 1: Y = Y + 1
    IF X = 0 THEN X = N
    IF Y > N THEN Y = 1
    IF A(X, Y) = 0 THEN
      A(X, Y) = FIRST + INC * (I - 1)
    ELSE
      X = X + 2: Y = Y - 1
      IF X > N THEN X = X - N
      IF Y = 0 THEN Y = N
      A(X, Y) = FIRST + INC * (I - 1)
    END IF
  NEXT I
RETURN
```

```
'3.8
' This program will display a pie graph.
'
DIM A(21, 21)
INPUT "Enter 3 percentages: "; P(1), P(2), P(3)
A$(1) = "A": A$(2) = "D": A$(3) = "N"
CLS : PI = 3.14159
' Draw circle
FOR I = -PI / 2 TO 3 / 2 * PI STEP .1
  X = COS(I) * 10:  Y = SIN(I) * 10
  LOCATE 11 + Y, 11 + X: PRINT "*": A(11 + Y, 11 + X) = 1
NEXT I
' Draw 3 line segments from center
FOR S = 0 TO 2
  SUM = SUM + P(S)
  I = -PI / 2 + 2 * PI * SUM / 100
  FOR R = 0 TO 10
    X = COS(I) * R: Y = SIN(I) * R
    LOCATE 11 + Y, 11 + X: PRINT "*": A(11 + Y, 11 + X) = 1
  NEXT R
NEXT S
A$ = INPUT$(1):  SUM = 0
' Fill regions with letters
FOR S = 1 TO 3
  LSUM = SUM:  SUM = SUM + P(S)
  FOR L = LSUM TO SUM
    I = -PI / 2 + 2 * PI * L / 100
    FOR R = 1 TO 9
      X = COS(I) * R: Y = SIN(I) * R
      IF A(11 + Y, 11 + X) = 0 THEN
        LOCATE 11 + Y, 11 + X: PRINT A$(S)
      END IF
    NEXT R
  NEXT L
NEXT S
```

```
'3.9
' This program produces a precedence of jobs to run.
'
INPUT "Enter number of dependencies:"; NUM
FOR I = 1 TO NUM
  INPUT "Enter dependency:"; DEP$: DEP$ = DEP$ + " "
  A$(I) = MID$(DEP$, 1, 3)
  B$(I) = MID$(DEP$, 4, 3)
  ' Store unique jobs in string
  IF INSTR(U$, A$(I)) = 0 THEN U$ = U$ + A$(I)
  IF INSTR(U$, B$(I)) = 0 THEN U$ = U$ + B$(I)
NEXT I
' Since there is a unique order for all the jobs,
' every job will have its successor somewhere in B().
' 1) search all B() for the only job missing.
' 2) exclude all dependencies with this job in it.
' 3) search all B() for the next only job missing.
' 4) repeat steps 2 and 3 until the final dependency is left.
L = LEN(U$): UNUM = L / 3: U2$ = U$: DEPLEFT = NUM:  JOBS$ = ""
WHILE DEPLEFT > 1
  FOR I = 1 TO NUM: MARKED(I) = 0: NEXT I
  FOR I = 1 TO NUM
    P = INSTR(U2$, B$(I))
    IF P > 0 THEN MARKED((P + 2) / 3) = -1
  NEXT I
  NOJOB = -1: I = 0
  WHILE NOJOB AND (I < UNUM)
    I = I + 1: ST = I * 3 - 2
    JOB$ = MID$(U2$, ST, 3)
    VALIDJOB = (INSTR(JOBS$, JOB$) = 0) AND (JOB$ <> SPACE$(3))
    IF VALIDJOB AND NOT MARKED(I) THEN
      JOBS$ = JOBS$ + JOB$
      FOR K = 1 TO NUM
        IF A$(K) = JOB$ THEN
          A$(K) = "*": B$(K) = "*"
          DEPLEFT = DEPLEFT - 1
        END IF
      NEXT K
      NEWU2$ = MID$(U2$, 1, ST - 1) + SPACE$(3)
      U2$ = NEWU2$ + MID$(U2$, ST + 3, L - ST - 2)
      NOJOB = 0
    END IF
  WEND
WEND
' Last dependency is concatenated
FOR I = 1 TO NUM
  IF A$(I) <> "*" THEN JOBS$ = JOBS$ + A$(I) + B$(I)
NEXT I
PRINT "JOBS MUST BE RUN IN THIS ORDER: "; JOBS$
```

```
'3.10
' This program finds a perfect square with digits 1-9.
'
DEFINT B, Z: DEFLNG A, N: MIN = 9
FOR NUM = 10001 TO INT(SQR(987654321))
  A = NUM * NUM
  DIGITS$ = LTRIM$(STR$(A))
  GOOD = -1: L = 1
  WHILE (L <= 9) AND GOOD
    IF INSTR(DIGITS$, CHR$(48 + L)) = 0 THEN GOOD = 0
    L = L + 1
  WEND
  IF GOOD THEN          'Found perfect square with unique digits
    GOSUB CheckDigits  'Count will contain number of swaps made
    IF COUNT < MIN THEN MIN = COUNT: NUMMIN = A: NUMMIN2 = NUM
  END IF
NEXT NUM
' Display the perfect square needing least number of swaps
DIGITS$ = LTRIM$(STR$(NUMMIN))
PRINT DIGITS$; " IS THE SQUARE OF"; NUMMIN2
PRINT "AND WAS FORMED BY EXCHANGING"; MIN; "PAIRS OF DIGITS"
END

CheckDigits:  'Determine number of swaps made and store in count
  FOR I = 1 TO 9: A(I) = VAL(MID$(DIGITS$, I, 1)): NEXT I
  COUNT = 0
  FOR I = 1 TO 9
    IF A(I) <> I THEN
      J = I + 1
      WHILE J < 9 AND A(J) <> I
        J = J + 1
      WEND
      SWAP A(I), A(J): COUNT = COUNT + 1
    END IF
  NEXT I
  RETURN
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '95 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T95;
{ -- This program displays title of contest forward/backward. }
  const
    A: String[50] =
       'FLORIDA HIGH SCHOOLS COMPUTING COMPETITION ''95';
  var
    I, J: Integer;

begin
  for I := 1 to 4 do begin
    Writeln (A);
    for J := Length(A) downto 1 do
      Write(Copy(A, J, 1));
    Writeln;
  end;
end.


{1.2}
program One2T95;
{ -- This program generates comments in different languages. }
  var
    C: String[60];

begin
  Write ('Enter comment: ');  Readln (C);
  Writeln ('BASIC: '' ', C);
  Writeln ('PASCAL: { ', C, ' }');
  Writeln ('C: /* ', C, ' */');
  Writeln ('C++: // ', C);
end.


{1.3}
program One3T95;
{ -- This program either increments or decrements N by 1. }
  var
    N:  Integer;
    Op: String[2];

begin
  Write ('Enter N: ');  Readln (N);
  Write ('Enter operator: ');  Readln (Op);
  if Op = '++' then
    Writeln (N + 1)
  else
    Writeln (N - 1);
end.
```

```
{1.4}
program One4T95;
{ -- This program rounds to three places by break point. }
  var
    BP:  Integer;
    Num: Real;

begin
  Write ('Enter break point: ');  Readln (BP);
  Write ('Enter number: ');  Readln (Num);
  Writeln ( Trunc((Num * 1000 + (10 - BP) / 10)) / 1000 :5:3);
end.


{1.5}
program One5T95;
{ -- This program determines if a program is a REXX or a CLIST. }
  var
    C: String[80];

begin
  Write ('Enter comment: ');  Readln (C);
  if Pos('REXX', C) > 0 then
    Writeln ('REXX')
  else
    Writeln ('CLIST');
end.


{1.6}
program One6T95;
{ -- This program displays the number of times variables appear.}
  var
    Num, Init, Init0: Integer;

begin
  Write ('Enter number of variables: ');  Readln (Num);
  Write ('Enter number initialized: ');   Readln (Init);
  Write ('Enter number initialized to 0: ');  Readln (Init0);
  Writeln ('BASIC = ', Init - Init0);
  Writeln ('PASCAL = ', Num + Init);
  Writeln ('C/C++ = ', Num);
end.
```

```
{1.7}
program One7T95;
{ -- This program displays last qualifier of a data set name. }
  var
    DSN:  String[44];
    Last: String[8];
    I:    Integer;
    Ch:   Char;

begin
  Write ('Enter data set name: ');  Readln (DSN);
  Last := '';
  for I := Length(DSN) downto 1 do begin
    Ch := DSN[I];
    if Ch = '.' then begin
      Writeln (Last); Exit; end
    else
      Last := Ch + Last;
  end;
end.
```

```
{1.8}
program One8T95;
{ -- This program displays real numbers in reverse order. }
  var
    I, N: Byte;
    A:    Array[1..10] of String[10];

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

```
{1.9}
program One9T95;
{ -- This program displays a large X made up of letter X's. }
uses Crt;
  var
    Num, I: Byte;

begin
  Write ('Enter number of X''s: ');  Readln (Num);
  ClrScr;
  for I := 1 to Num do begin
    GotoXY (I, I);  Write ('X');
    GotoXY (Num - I + 1, I);  Write ('X');
  end;
end.
```

```
{1.10}
program One10T95;
{ -- This program will display the savings in postage. }
  const
    Cost = 23.33333;
  var
    PS, SS, Oz1, Oz2, Page1, Page2: Integer;

begin
  Write ('Enter # of printed sides: ');  Readln (PS);
  Write ('Enter # of single sided pages: ');  Readln (SS);
  { -- Calculate # of pages and wieght for 1st bill }
  Page1 := PS - 6;  Oz1 := 1;
  Oz1 := Oz1 + (Page1 + 8) div 9;
  { -- Calculate # of pages and wight for 2nd bill }
  Page2 := SS + ((PS - SS + 1) div 2) - 6;
  Oz2 := 1;
  Oz2 := Oz2 + (Page2 + 8) div 9;
  Writeln ((Oz1 - Oz2) * Cost :6:2, ' CENTS SAVED');
end.
```

```
{2.1}
program Two1T95;
{ -- This program finds integral solutions of (X,Y) for AX+BY=C }
var
  A, B, C, X: Integer;
  Y:          Real;

begin
  Write ('Enter A, B, C: ');  Readln (A, B, C);
  X := 1;
  repeat
    Y := (C - A * X) / B;
    if Abs(Y - Trunc(Y)) < 0.001 then begin
      Writeln ('(', X, ',', Y :1:0, ')');  Exit;
    end;
    Inc(X);
  until X > 10000;
end.
```

```
{2.2}
program Two2T95;

{ -- This program verifies a number by validating check digit. }
  var
    Part:             String[20];
    Prod, Sum, Code: Integer;
    I, L, Digit, ChkDigit, LastDigit: Byte;

begin
  Write ('Enter part number: ');  Readln (Part);
  L := Length(Part);  Prod := 1;
  for I := 1 to L - 1 do begin
    Val(Copy(Part, I, 1), Digit, Code);
    Sum := Sum + Digit * ((I mod 2) + 1);
  end;
  { -- Subtract units digit of Sum from 9 for check digit }
  ChkDigit := 9 - (Sum mod 10);
  Val(Copy(Part, L, 1), LastDigit, Code);
  if ChkDigit = LastDigit then
    Writeln ('OKAY')
  else
    Writeln ('ERROR -- CHECK DIGIT SHOULD BE ', ChkDigit);
end.
```

```
{2.3}
program Two3T95;
{ -- This program determines # of prizes given of $13 million. }
  var
    Prize: LongInt;
    Pow:   Array[0..7] of LongInt;
    A:     Array[0..6] of Byte;
    I:     Byte;

begin
  Prize := 13000000;
  { -- Same algorithm is used as converting # to base 13 #. }
  Pow[7] := 1;
  for I := 1 to 7 do Pow[7] := Pow[7] * 13;
  for I := 6 downto 0 do begin
    Pow[I] := Pow[I+1] div 13;
    A[I]   := Prize div Pow[I];
    Prize  := Prize mod Pow[I];
  end;
  for I := 0 to 6 do
    Writeln ('$', Pow[I], ' = ', A[I]);
end.
```

```
{2.4}
program Two4T95;
{ -- This program determines the cost of Directory Assistance. }
  var
    DAC, Area:      String[11];
    I, N, LocalDAC: Byte;
    Tot, Cost:      Real;

begin
  Write ('Enter number of DACs: ');  Readln (N);
  for I := 1 to N do begin
    Write ('Enter DAC: ');  Readln (DAC);
    if DAC = '00' then
      Cost := 3.00
    else if DAC = '1411' then begin
      Inc(LocalDAC);  Cost := 0;  end
    else begin
      Area := Copy(DAC, 2, 3);
      if Area = '813' then
        Cost := 0.25
      else
        if (Area = '305') or (Area = '407') or (Area = '904') then
          Cost := 0.40
        else
          Cost := 0.65;
    end;
    Tot := Tot + Cost;
  end;  { -- for I }
  { -- Every local DAC after the third cost 25 cents }
  if LocalDAC > 3 then
    Tot := Tot + (LocalDAC - 3) * 0.25;
  Writeln (Tot: 5:2, ' DOLLARS');
end.
```

```
{2.5}
program Two5T95;
{ -- This program will display the heading of even/odd pages. }
  const
    PNum: Array [1..4] of Integer = (180, 140, 200, 260);
    P:    Array [1..4] of String[17] =
           ('PROBLEMS', 'JUDGING CRITERIA',
            'BASIC SOLUTIONS', 'PASCAL SOLUTIONS');
  var
    I, Pag, Page, Chapter: Integer;

begin
  Write ('Enter page number: ');  Readln (Page);
  if Page mod 2 = 0 then begin
    Write (Page, '  FLORIDA HIGH SCHOOLS COMPUTING COMPETITION');
    Writeln (' 1985 - 1994');
    end
  else begin
    Write ('FHSCC ''');
    I := 1;  Pag := Page;
    while Pag > PNum[I] do begin
      Pag := Pag - PNum[I];  Inc(I);
    end;
    Chapter := Trunc(Pag / (PNum[I] / 10));
    Writeln (85 + Chapter, ' ', P[I], '  ', Page);
  end;
end.
```

```
{2.6}
program Two6T95;
{ -- This program computes total ESTIMATED PREPARATION TIME. }
  const
    Form: Array[1..6] of String[4] = ('1040','A','B','C','D','E');
    Hr  : Array[1..6,1..4] of Integer = ((3,2,4,0), (2,0,1,0),
          (0,0,0,0), (6,1,2,0), (0,0,1,0), (2,1,1,0));
    Min : Array[1..6,1..4] of Integer = ((8,53,41,53),
          (32,26,10,27), (33,8,17,20), (26,10,5,35),
          (51,42,1,41), (52,7,16,35));
  var
    I, J, TotHr, TotMin: Integer;
    F: String[4];

begin
  I := 0;
  repeat
    Write ('Enter form: ');  Readln (F);
    I := 1;
    while (I < 7) and (F <> Form[I]) do Inc(I);
    if I < 7 then
      for J := 1 to 4 do begin
        Inc(TotHr, Hr[I,J]);
        Inc(TotMin, Min[I,J]);

      end;
  until I > 6;

  Inc(TotHr, TotMin div 60);
  TotMin := TotMin mod 60;
  Writeln (TotHr, ' HR., ', TotMin, ' MIN.');
end.
```

```
{2.7}
program Two7T95;
{ -- This program will calculate investments at GTE. }
  const
    BegPrice: Real = 27.20;
    Return401K: Real = 0.14;
  var
    Salary, Percent, EndPrice, StockGain: Real;
    CompCont, EmpCont, K401, TotalGain:   Real;
    MaxShares, Shares:                    Integer;

begin
  Write ('Enter salary: ');  Readln (Salary);
  Write ('Enter 401K %: ');  Readln (Percent);
  Percent := Percent / 100;
  MaxShares := Trunc(Salary / 100);
  Writeln ('YOU CAN PURCHASE UP TO ', MaxShares, ' SHARES');
  Write ('Enter number of shares: ');  Readln (Shares);
  Write ('Enter end of year price: '); Readln (EndPrice);

  EmpCont := Salary * Percent;
  if Percent >= 0.06 then
    CompCont := (Salary * 0.06) * 0.75
  else
    CompCont := (Salary * Percent) * 0.75;
  K401 := (EmpCont + CompCont) * Return401K;
  StockGain := Shares * (EndPrice - BegPrice);
  TotalGain := CompCont + K401 + StockGain;

  Writeln ('COMPANY CONTRIBUTION: ', CompCont  :8:2);
  Writeln ('         401K RETURN: ', K401      :8:2);
  Writeln ('          STOCK GAIN: ', StockGain :8:2);
  Writeln ('          TOTAL GAIN: ', TotalGain :8:2);
end.
```

```
{2.8}
program Two8T95;
{ -- This program will produce loops of a spiral using letters. }
uses Crt;
  var
    Num, Row, Col, Incr, LoopNum, I: Byte;
    Let: Char;

begin
  Write ('Enter number of spiral loops: ');  Readln (Num);
  Write ('Enter first letter: ');  Readln (Let);
  ClrScr;
  Row := 12;  Col := 40;  Incr := 1;
  while LoopNum < Num do begin
    Incr := Incr + 2;
    { -- Go right }
    GotoXY (Col, Row);  for I := 1 to Incr do Write (Let);
    Col := Col + Incr - 1;
    { -- Go down }
    for I := 1 to Incr - 1 do begin
      GotoXY (Col, Row + I);  Write (Let);
    end;
    Row := Row + Incr - 1;  Incr := Incr + 2;
    { -- Go left }
    Col := Col - Incr + 1;
    GotoXY (Col, Row);  for I := 1 to Incr do Write (Let);
    { -- Go up }
    for I := 1 to Incr - 2 do begin
      GotoXY (Col, Row - I);  Write (Let);
    end;
    Row := Row - Incr + 1;
    if Let = 'Z' then
      Let := 'A'
    else
      Let := Chr(Ord(Let) + 1);
    Inc(LoopNum);
  end;
end.
```

```
{2.9}
program Two9T95;
{ -- This program shows all possible moves for a Queen in chess.}
uses Crt;
  var
    Col, Row, I, J, Code: Integer;
    RC:    String[2];
    R, C:  Array[1..4] of Integer;

begin
  Write ('Enter column and row: ');  Readln (RC);
  Col := Ord(RC[1]) - Ord('A') + 1;
  Val(Copy(RC, 2, 1), Row, Code);
  Row := 9 - Row;
  ClrScr;
  for I := 8 downto 1 do Writeln (I);
  Writeln ('  A B C D E F G H');
  { -- Horizontal moves }
  GotoXY (3, Row);  Writeln ('* * * * * * * *');
  { -- Vertical moves }
  for I := 1 to 8 do begin
    GotoXY (Col * 2 + 1, I);  Write ('*');
  end;
  { -- Diagonal moves }
  for I := 1 to 7 do begin
    R[1] := Row - I;  C[1] := Col - I;
    R[2] := Row + I;  C[2] := Col + I;
    R[3] := Row - I;  C[3] := Col + I;
    R[4] := Row + I;  C[4] := Col - I;
    for J := 1 to 4 do
      if (R[J] > 0) and (R[J] < 9) and (C[J] > 0) and (C[J] < 9)
      then begin
        GotoXY (C[J] * 2 + 1, R[J]);  Write ('*');
      end;
  end;
  GotoXY (Col * 2 + 1, Row);  Write('Q');
end.
```

```
{2.10}
program Two10T95;
{ -- This program tabulates information during a pre-election. }
  const
    A: Array[1..10] of String[37] = ('MALE', 'FEMALE',
       '50 AND BELOW', 'OVER 50', 'WHITE', 'OTHERS',
       'ABOVE $25000', '$25000 AND BELOW',
       'WHITE MALE OVER 50 AND ABOVE $25000', 'OTHER');
  var
    Sex, Race, Party:         Char;
    Income:                   LongInt;
    Row, Col, Age, Total:     Byte;
    Sum: Array[1..10,1..2] of Byte;

begin
  Total := 0;
  for Row := 1 to 10 do
    for Col := 1 to 2 do
      Sum[Row, Col] := 0;
  Write ('Enter sex: ');  Readln (Sex);
  while (Sex <> 'E') do begin
    Write ('Enter age: ');    Readln (Age);
    Write ('Enter race: ');   Readln (Race);
    Write ('Enter income: '); Readln (Income);
    Write ('Enter party: ');  Readln (Party);
    if Party = 'D' then Col := 1 else Col := 2;
    if Sex = 'M'   then Row := 1 else Row := 2;
    Inc(Sum[Row,Col]);
    if Age <= 50   then Row := 3 else Row := 4;
    Inc(Sum[Row,Col]);
    if Race = 'W'  then Row := 5 else Row := 6;
    Inc(Sum[Row,Col]);
    if Income > 25000 then Row := 7 else Row := 8;
    Inc(Sum[Row,Col]);
    if (Race = 'W') and (Sex = 'M') and (Age > 50) and (Row = 7)
      then Row := 9 else Row := 10;
    Inc(Sum[Row,Col]);
    Inc(Total);
    Writeln;
    Write ('Enter sex: ');  Readln (Sex);
  end;
  Write (' ':32, 'DEMOCRATIC  REPUBLICAN');
  for Row := 1 to 10 do begin
    if Row mod 2 = 1 then Writeln;
    Write (A[Row], ' ': 37 - Length(A[Row]));
    Write (Sum[Row, 1] / Total * 100 :5:1);
    Writeln (' ':7, Sum[Row,2] / Total * 100 :5:1);
  end;
end.
```

```
{3.1}
program Thr1T95;
{ -- This program will determine how much IRS owes/pays. }
  const
    Amount:    Array[0..5] of Real =
               (0, 22750, 55100, 115000, 250000, 9999999);
    Rate:      Array[0..5] of Real =
               (0, 0.15, 0.28, 0.31, 0.36, 0.396);
    StDeduct:  Real = 3800;
    Exemption: Real = 2450;
  var
    Gross, Deductions, FedTax, Income, TaxInc, Tax: Real;
    I, J: Byte;

begin
    Write ('Enter adjusted gross income: ');  Readln (Gross);
    Write ('Enter itemized deductions: ');  Readln (Deductions);
    Write ('Enter federal income tax withheld: ');
    Readln (FedTax);
    if Deductions > StDeduct then
      Income := Gross - Deductions
    else
      Income := Gross - StDeduct;
    TaxInc := Income - Exemption;

    Tax := 0;
    for I := 1 to 5 do
      if TaxInc <= Amount[I] then begin
        for J := 1 to I - 1 do
          Tax := Tax + (Amount[J] - Amount[J-1]) * Rate[J];
        Tax := Tax + (TaxInc - Amount[I-1]) * Rate[I];
        Write (Abs(Tax - FedTax) :9:2, ' DOLLARS ');
        if FedTax < Tax then
          Writeln ('YOU OWE')
        else
          Writeln ('WILL BE REFUNDED TO YOU');
        Exit;
      end;
end.
```

```
{3.2}
program Thr2T95;
{ -- This program will display a simplified phone bill. }
  var
    I, L, HH, Code:          Integer;
    Rate1, Rate2, Tot, Disc: Real;
    Min:     Array[1..10] of Byte;
    Tim:     Array[1..10] of String[13];
    Charge:  Array[1..10] of Real;
    AM, Day: String[3];
    Midday:  Boolean;
```

```
begin
  L := 1;  Tot := 0;
  Write ('Enter MIN: ');  Readln (Min[L]);
  while Min[L] > 0 do begin
    Write ('Enter time: ');  Readln (Tim[L]);
    Inc(L);
    Write ('Enter MIN: ');   Readln (Min[L]);
  end;
  Dec(L);
  { -- Display bill }
  Writeln ('  BOB SMITH  (813) 555-1234');  Writeln;
  Writeln ('  TIME OF DAY  MIN.  CHARGE');
  for I := 1 to L do begin
    if Copy(Tim[I], 1, 1) = '0' then
      Write (' ', Copy (Tim[I], 2, 12))
    else
      Write (Tim[I]);
    { -- Calculate charge }
    Val(Copy(Tim[I], 1, 2), HH, Code);
    AM := Copy(Tim[I], 7, 2);
    Day := Copy(Tim[I], 11, 3);
    Midday := (  (HH > 7)  and (HH < 12) and (AM = 'AM')
              or (HH = 12) and (AM = 'PM')
              or (HH < 5)  and (AM = 'PM') );
    if (HH > 4) and (HH < 11) and (AM = 'PM') and (Day <> 'SAT')
    then
      begin
        Rate1 := 0.21;  Rate2 := 0.16;
      end
    else if Midday and (Day <> 'SAT') and (Day <> 'SUN') then
      begin
        Rate1 := 0.28;  Rate2 := 0.21;
      end
    else
      begin
        Rate1 := 0.14;  Rate2 := 0.11;
      end;
    Charge[I] := Rate1 + Rate2 * (Min[I] - 1);
    Writeln (Min[I] :5, '   ', Charge[I]: 6:2);
    Tot := Tot + Charge[I];
  end;
  if Tot > 20 then Disc := Tot * 0.20;
  Writeln;
  Writeln ('TOTAL CHARGES', ' ': 8,  Tot: 6:2);
  Writeln ('DISCOUNT',      ' ': 13, Disc: 6:2);
  Writeln ('CHARGES - DISCOUNT   ',  Tot - Disc :6:2);
end.
```

```
{3.3}
program Thr3T95;
{ -- 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.
```

```
{3.4}
program Thr4T95;
{ -- This program will produce all possible subsets of letters. }
  var
    Sub:       Array[1..1024] of String[10];
    Let, XSub: String[10];
    A:         Array[1..10] of Char;
    X:         Char;
    I, J, L, Col, SubLen, Bit: Byte;
    N, Num, Two2L, Power:      Integer;

begin
  Write ('Enter letters: ');  Readln (Let);
  L := Length(Let);
  for I := 1 to L do A[I] := Let[I];
  { -- Sort letters in A[] }
  for I := 1 to L - 1 do
    for J := I + 1 to L do
      if A[I] > A[J] then begin
        X := A[I]; A[I] := A[J]; A[J] := X;
      end;
  { -- Generate binary numbers to produce all subsets }
  Two2L := 1;
  for I := 1 to L do Two2L := Two2L * 2;
  for N := 0 to Two2L - 1 do begin
    Num := N; Power := Two2L;  Sub[N] := '';
    for J := L - 1 downto 0 do begin
      Power := Power div 2;
      Bit := Num div Power;
      if Bit = 1 then begin
        Sub[N] := Sub[N] + A[L - J];  Num := Num - Power;
      end;
    end;
  end;
  { -- Bubble sort subsets }
  for I := 0 to Two2L - 2 do
    for J := I + 1 to Two2L - 1 do
      if Sub[I] > Sub[J] then begin
        XSub := Sub[I];  Sub[I] := Sub[J];  Sub[J] := XSub;
      end;
  { -- Display subsets }
  Col := 0;
  for I := 0 to Two2L - 1 do begin
    SubLen := Length(Sub[I]) + 3;
    if Col + SubLen > 50 then begin
      Writeln;  Col := 0;
    end;
    Write ('{', Sub[I], '} ');
    Col := Col + SubLen;
  end;
  Writeln;  Writeln('TOTAL SUBSETS = ', Two2L);
end.
```

```
{3.5}
program Thr5T95;
{ -- This program will sum big integers from 1 to N. }
{ -- Gauss's formula: SUM = N * (N+1) / 2.           }
  var
    A, B, Prod, D:              Array[1..80] of Byte;
    I, J, S, Carry, LenA, LenB: Byte;
    N:                          String[40];
    Code:                       Integer;

begin
  Write ('Enter N: ');  Readln (N);
  { -- Store digits of N in A[] and B[] }
  LenA := Length(N);  LenB := LenA;
  for I := 1 to LenA do begin
    Val(Copy(N, LenA - I + 1, 1), A[I], Code);
    B[I] := A[I];
  end;
  { -- Add 1 to number in B[] }
  Inc(B[1]);  I := 1;
  while (B[I] = 10) do begin
    B[I] := 0;  Inc(I);  Inc(B[I]);
  end;
  if I > LenB then LenB := I;
  { -- Multiply A[] by B[] }
  for I := 1 to LenA do begin
    Carry := 0;
    for J := 1 to LenB do begin
      S := I + J - 1;
      Prod[S] := Prod[S] + A[I] * B[J] + Carry;
      Carry := Prod[S] div 10;
      Prod[S] := Prod[S] - Carry * 10;
    end;
    if Carry > 0 then Prod[S+1] := Carry;
  end;
  if Carry > 0 then Inc(S);
  { -- Divide product Prod[] by 2 }
  if Prod[S] = 1 then begin
    Dec(S);  Carry := 10;
  end;
  for I := S downto 1 do begin
    D[I] := (Prod[I] + Carry) div 2;
    Carry := (Prod[I] mod 2) * 10;
  end;
  { -- Display answer in D[] }
  for I := S downto 1 do  Write (D[I]);
  Writeln;
end.
```

```
{3.6}
program Thr6T95;
{ -- This program will assign values to variables in BASIC code.}
  var
    L, I, PosV, PosV2, PosV3, Num1, Num2, Code: Integer;
    A: Array[1..12] of String[5];
    B: Array[1..12] of Integer;
    V, Ch, Op: Char;
    AllV: String[5];

begin
  L := 0;
  repeat
    Inc(L);
    Write ('Enter line: ');  Readln (A[L]);
  until A[L] = 'END';
  Dec(L);

  AllV := '';
  for I := 1 to L do begin
    { -- Determine if first variable is new or old }
    V := A[I,1];
    PosV := Pos(V, AllV);
    if PosV = 0 then begin
      AllV := AllV + V;
      PosV := Length(AllV);
    end;
    { -- Assign value for first number }
    Ch := A[I,3];
    if (Ch in ['0'..'9']) then
      Val(Ch, Num1, Code)
    else begin
      PosV2 := Pos(Ch, AllV);
      Num1 := B[PosV2];
    end;

    if Length(A[I]) = 3 then
      { -- Assign first number to current variable }
      B[PosV] := Num1
    else begin
      { -- Assign value for second number }
      Ch := A[I,5];
      if Ch in ['0'..'9'] then
        Val(Ch, Num2, Code)
      else begin
        PosV3 := Pos(Ch, AllV);
        Num2 := B[PosV3];
      end;
      { -- Perform operation with 1st and 2nd num, place in var }
      Op := A[I,4];
      Case Op of
        '+': B[PosV] := Num1 + Num2;
        '-': B[PosV] := Num1 - Num2;
        '*': B[PosV] := Num1 * Num2;
        '/': B[PosV] := Num1 div Num2;
      end;
    end;
  end;  { -- for I }
  { -- Display the variables in order of appearance with values }
  for I := 1 to Length(AllV) do
    Writeln (Copy(AllV, I, 1), '=', B[I]);
end.
```

```
{3.7}
program Thr7T95;
{ -- This program finds three 3-digit primes having digits 1-9. }
  var
    A:           Array[1..200] of LongInt;
    Digits:      String[9];
    Prime, Good: Boolean;
    I, J, K, L, H, T, One, P, Sum, PNum: Integer;
begin      { -- Generate primes into A[] }
  P := 0; I := 101;
  repeat
    J := 3;  Prime := True;
    while (J <= Sqrt(I)) and Prime do begin
      if I mod J = 0 then Prime := False;
      J := J + 2;
    end;
    if prime then begin
      { -- Ensure that Digits are unique and not 0 }
      H := I div 100;  T := (I - H * 100) div 10;
      One := I - H * 100 - T * 10;
      if (T > 0) and (H <> T) and (T <> One) and (H <> One) then
        begin  Inc(P);  A[P] := I;  end;
    end;
    Inc(I, 2);
  until I > 997;
  { -- Add the different combinations of 3 primes }
  for I := 1 to P - 2 do
    for J := I + 1 to P - 1 do
      for K := J + 1 to P do begin
        Sum := A[I] + A[J] + A[K];
        { -- Check if Sum has 4 digits in ascending order }
        if Sum >= 1234 then begin
          Str(Sum, Digits);  Good := True;  L := 1;
          repeat
            if Digits[L] >= Digits[L+1] then Good := False;
            Inc(L);
          until (L = 4) or not Good;
          { -- Check all 3-digit primes for digits 1 through 9 }
          if Good then begin
            Str((((A[I] * 1000 + A[J]) * 1000) + A[K]), Digits);
            L := 1;
            while (L <= 9) and Good do begin
              if Pos(Chr(48+L), Digits) = 0 then Good := False;

              Inc(L);
            end;
            if Good then begin
              Writeln (A[I],' + ',A[J],' + ',A[K],' = ', Sum);
              Inc(PNum);  If PNum = 7 then Exit;
            end;
          end;
        end;
      end;  { -- for K }
end.
```

```
{3.8}
program Thr8T95;
{ -- This program will display time in MM:SS in block letters. }
uses Crt;
  const
    B: Array[1..5] of String[60] = (
    '****     *  ****  ****  *  *  ****  *     ****  ****  ****',
    '*  *     *     *     *  *  *  *     *        *  *  *  *  *',
    '*  *     *  ****  ****  ****  ****  ****     *  ****  ****',
    '*  *     *  *        *     *     *  *  *     *  *  *     *',
    '****     *  ****  ****     *  ****  ****     *  ****     *'
    );
    { -- Maximum units for MM:SS }
    Max: Array[1..4] of Byte = (6, 10, 6, 10);
    { -- Columns to start blocks }
    Col: Array[1..4] of Byte = (1, 7, 18, 24);
  var
    I, J: Byte;
    Dig:  Array[0..9] of Byte;
    A:    Array[1..5,0..9] of String[4];
    MMSS: String[5];
    Code: Integer;
    Ch:   String[1];

begin
  for I := 1 to 5 do
    for J := 0 to 10 do
      A[I,J] := Copy(B[I], J * 6 + 1, 4);
  Write ('Enter MM:SS: '); Readln (MMSS);
  for I := 1 to 4 do
    if I < 3 then
      Val(Copy(MMSS, I, 1), Dig[I], Code)
    else
      Val(Copy(MMSS, I+1, 1), Dig[I], Code);

  ClrScr;
  GotoXY (14,2); Write('*');  GotoXY (14,4);  Write('*');
  Ch := '';
  repeat
    for I := 1 to 4 do
      for J := 1 to 5 do begin
        GotoXY (Col[I], J);  Write (A[J, Dig[I]]);
      end;
    Inc(Dig[4]);
    for J := 4 downto 1 do
      if Dig[J] = Max[J] then begin
        Inc(Dig[J-1]);  Dig[J] := 0;
      end;
    Delay(1000);
    if KeyPressed then Ch := ReadKey;
  until Ch <> ''
end.
```

```
{3.9}
program Thr9T95;
{ -- 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 Thr10T95;
{ -- This program displays versions of libraries on a graph. }
uses Crt;
  var
    Vers, FirstWk, FWkDisp, WkNum, LWkDisp, LastWk, Backup,
    I, Min, Max, TestArea, FirstPreWk, LastPreWk: Integer;

begin
  Write ('Enter version #: ');  Readln (Vers);
  Write ('Enter first week in test: ');  Readln (FirstWk);
  Write ('Enter first week to display, # of weeks: ');
  Readln (FWKDisp, WkNum);
  ClrScr;
  LWkDisp := FWkDisp + WkNum - 1;
  { -- Display week #s at top (units first, then tens) }
  Write (' ': 9);
  for I := FWkDisp to LWkDisp do  Write (I div 10);
  Writeln;  Write (' ': 9);
  for I := FWkDisp to LWkDisp do  Write (I mod 10);
  Writeln;  Writeln;
  LastWk := FirstWk + 17;
  { -- Compute # of versions to backup from Vers input }
  Backup := (LastWk - FWkDisp) div 6;
  Vers := Vers - Backup;
  FirstWk := FirstWk - 6 * Backup;
  LastWk  := LastWk  - 6 * Backup;
  repeat
    { -- Display Version and indent }
    Write ('R1V');  if Vers < 10 then Write ('0');
    Write(Vers, 'L01 ');
    if FWkDisp <= FirstWk then begin
      Min := FirstWk;
      Write (' ': FirstWk - FWkDisp); end
    else
      Min := FWkDisp;
    if LWkDisp >= LastWk then Max := LastWk else Max := LWkDisp;
    { -- Display TestArea of 1 if Vers even, 2 if odd; P = Prod }
    TestArea := (Vers mod 2) + 1;
    for I := Min to Max do
      if I < FirstWk + 12 then
        Write (TestArea)
      else
        Write ('P');
    Writeln;
    { -- Display Pre-Production Version }
    FirstPreWk := FirstWk + 5;  LastPreWk := FirstWk + 10;
    if (LastPreWk >= FWkDisp) and (FirstPreWk <= LWkDisp) then
      begin
        Write ('R1V');  if Vers - 1 < 10 then Write ('0');
        Write (Vers - 1, 'L88 ');
        if FirstPreWk > FWkDisp then begin
          Min := FirstPreWk;

          Write (' ': FirstPreWk - FWkDisp); end
        else
          Min := FWkDisp;
        if LWkDisp >= LastPreWk then
          Max := LastPreWk
        else
          Max := LWkDisp;
        for I := 1 to Max - Min + 1 do Write ('*');
        Writeln;
      end; { -- if }
    FirstWk := FirstWk + 6;  LastWk := LastWk + 6;
    Inc(Vers);
  until FirstWk > LWkDisp;
end.
```

```
{ -- FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '96 }
{ -- PASCAL PROGRAM SOLUTIONS }


{1.1}
program One1T96;
{ -- This program displays a phrase of the form FHSCC '##. }
  var
    Year: String[4];

begin
  Write ('Enter year: ');  Readln (Year);
  Writeln ('FHSCC ''', Copy(Year,3,2));
end.


{1.2}
program One2T96;
{ -- This program tallies number of frequent flier miles. }
  var
    X, Y: Integer;

begin
  Write ('Enter X: ');  Readln (X);
  Write ('Enter Y: ');  Readln (Y);
  Writeln (X * (1300 + 1300 + 500) + (Y * 5));
end.


{1.3}
program One3T96;
{ -- This program displays middle letter(s) of a word. }
  var
    Word: String[20];
    L, M: Integer;

begin
  Write ('Enter word: ');  Readln (Word);
  L := Length(Word);  M := L div 2;
  If (L mod 2) = 0 then Write (Copy(Word, M, 1));
  Writeln (Copy(Word, M+1, 1));
end.
```

```
{1.4}
program One4T96;
{ -- This program displays area and perimeter of a rectangle. }
  var
    X1, Y1, X2, Y2, Area, Perim: Integer;

begin
  Write ('Enter coordinate 1: ');  Readln (X1, Y1);
  Write ('Enter coordinate 2: ');  Readln (X2, Y2);
  Area  := Abs((X1 - X2) * (Y1 - Y2));
  Perim := (Abs(X1 - X2) + Abs(Y1 - Y2)) * 2;
  Writeln ('AREA = ', Area);
  Writeln ('PERIMETER = ', Perim);
end.
```

```
{1.5}
program One5T96;
{ -- This program code-breaks an encrypted secret message. }
  var
    E: String[40];
    M: Char;
    I: Integer;

begin
  Write ('Enter encryption: ');  Readln (E);
  for I := 1 to Length(E) do begin
    M := E[I];
    if M = ' ' then
      Write(M)
    else
      Write (Chr( Ord('Z') - Ord(M) + Ord('A') ));
  end;
  Writeln;
end.
```

```
{1.6}
program One6T96;
{ -- This program displays number of floors elevator touches. }
  var
    Floor, Total, Max, LastFloor: Integer;

begin
  repeat
    Write ('Enter floor: ');  Readln (Floor);
    Total := Total + Abs(Floor - LastFloor);
    if Floor > Max then Max := Floor;
    LastFloor := Floor;
  until (Floor = 0);
  { -- 1 is added for the starting ground floor }
  Writeln ('TOTAL FLOORS TOUCHED = ', Total + 1);
  Writeln ('UNIQUE FLOORS TOUCHED = ', Max + 1);
end.
```

```
{1.7}
program One7T96;
{ -- This program displays a person's ratios for buying a house.}
  var
    Loan, Debts, Income, Ratio1, Ratio2: Real;

begin
  Write ('Enter amount of loan: ');   Readln (Loan);
  Write ('Enter amount of debts: ');  Readln (Debts);
  Write ('Enter amount of income: '); Readln (Income);
  Ratio1 := (Loan / Income) * 100;
  Ratio2 := ((Loan + Debts) / Income) * 100;
  Writeln ('RATIOS = ', Ratio1: 4:1, '% / ', Ratio2: 4:1, '%');
  Write   ('DOES ');
  if (Ratio1 > 33) or (Ratio2 > 38) then Write ('NOT ');
  Writeln ('QUALIFY');
end.
```

```
{1.8}
program One8T96;
{ -- This program will convert numbers to English or Spanish.}
  const
    N: Array [1..20] of String[6] = ('ONE','TWO','THREE',
    'FOUR','FIVE','SIX','SEVEN','EIGHT','NINE','TEN',
    'UNO','DOS','TRES','CUATRO','CINCO','SEIS','SIETE',
    'OCHO','NUEVE','DIEZ');
  var
    Lang:   Char;
    Num, I: Byte;

begin
  Write ('Enter E or S: ');  Readln (Lang);
  Write ('Enter number: ' ); Readln (Num);
  if Lang = 'S' then I := 10 else I := 0;
  Writeln (N[I + Num]);
end.
```

```
{1.9}
program One9T96;
{ -- This program forms a cross from word(s). }
  var
    W: String[20];
    I, L, M: Byte;

begin
  Write ('Enter word(s): ');  Readln (W);
  L := Length(W);  M := (L div 2) + 1;
  for I := 1 to L do
    If I <> M then
      Writeln (' ': M - 1, Copy(W, I, 1))
    else
      Writeln (W);
end.
```

```
{1.10}
program One10T96;
{ -- This program simulates the PRICE IS RIGHT game. }
  var
    Price, Min, I, Dif, Index: Integer;
    A:     Array[1..4] of Integer;

begin
  Write ('Enter actual price: ');  Readln (Price);
  Write ('Enter guesses A, B, C, D: ');
  Readln (A[1], A[2], A[3], A[4]);
  Min := 32000;
  for I := 1 to 4 do
    if A[I] <= Price then begin
      Dif := Price - A[I];
      if Dif < Min then begin
        Min := Dif;  Index := I;
      end;
    end;
  if Index > 0 then
    Writeln ('PERSON ', Copy ('ABCD', Index, 1))
  else
    Writeln ('EVERYONE IS OVER');
end.
```

```
{2.1}
program Two1T96;
{ -- This program will emulate random dart throws. }
  const
    S: Array[1..7] of Byte = (0,2,4,5,10,20,50);
  var
    X, Throw, Total: Byte;

begin
  Randomize;  Throw := 0;
  repeat
    X := Random(7) + 1;  Inc(Throw);
    Write(S[X]);
    Inc(Total, S[X]);
    If Total < 100 then Write (',');
  until (Total >= 100);
  Writeln;
  Writeln (Throw, ' THROWS ACHIEVED A SCORE OF ', Total);
  Writeln;
end.
```

```
{2.2}
program Two2T96;
{ -- This program compresses information to save space. }
  var
    S:      String[80];
    I, Ast: Byte;
    Md:     Char;

begin
  Write ('Enter string: ');  Readln (S);
  Ast := 0;
  for I := 1 to Length(S) do begin
    Md := S[I];
    if Md <> '*' then
      begin
        if Ast > 0 then
          begin
            if Ast = 1 then
              Write ('*')
            else
              Write (Ast);
            Ast := 0;
          end;
        Write(Md);
      end
    else
      Inc(Ast)
  end;  { -- for I }
  Writeln;
end.
```

```
{2.3}
program Two3T96;
{ -- This program finds 2 numbers to add to the set 1,3,8. }
  var
    A:             Array[1..5] of Integer;
    I, J, Num, N: Integer;
    Found:         Boolean;

begin
  A[1] := 1; A[2] := 3; A[3] := 8;  N := 3;  I := 0;
  for I := 0 to 999 do begin
    Found := True;
    for J := 1 to N do begin
      Num := A[J] * I + 1;
      if Sqrt(Num) - Trunc(Sqrt(Num + 0.0001)) > 0.0001 then
        Found := False;
    end;
    if Found then begin
      Writeln (I);  Inc(N);  A[N] := I;  if N = 5 then Exit;
    end;
  end;
end.
```

```
{2.4}
program Two4T96;
{ -- This program displays the LCM of the first N integers. }
  var
    A:       Array[1..31] of Integer;
    I, J, N: Integer;
    Prod:    Real;

begin
  Write ('Enter N: ');  Readln (N);
  for I := 2 to N do  A[I] := I;
  { -- Produce all the necessary prime factors. }
  for I := 2 to N do
    for J := I + 1 to N do
      if (A[J] Mod A[I]) = 0 then A[J] := A[J] div A[I];

  Prod := 1;
  For I := 2 to N do
    Prod := Prod * A[I];

  Writeln (Prod: 13:0);
end.
```

```
{2.5}
program Two5T96;
{ -- This program will calculate the fractional value. }
  var
    Word:    String[3];
    A:       Array[1..3] of Integer;
    I, N, D: Integer;

begin
  Write ('Enter word: ');  Readln (Word);
  for I := 1 to 3 do
    A[I] := Ord(Word[I]) - Ord('A') + 1;

  N := A[1] * A[2] + A[2] * A[3] + A[1] * A[3];
  D := A[1] * A[2] * A[3];
  for I := D downto 1 do
    if (N mod I = 0) and (D mod I = 0) then begin
      Writeln (N div I, '/', D div I);  Exit
    end;
end.
```

```
{2.6}
program Two6T96;
{ -- This program displays the Nth prime in Fibonacci sequence. }
  var
    F:             Array[1..99] of LongInt;
    I, N, J, PNum: Integer;
    Prime:         Boolean;

begin
  F[1] := 1;  F[2] := 1;  F[3] := 2;  PNum := 1;  I := 3;
  Write ('Enter N: ');  Readln (N);
  while (PNum < N) do begin
    Inc(I);
    F[I] := F[I-1] + F[I-2];  Prime := True;
    { -- Check if Fibonacci # is prime (not divis by 2 or odd#) }
    if (F[I] Mod 2 = 0) then Prime := False;
    if Prime then begin
      for J := 3 to Trunc(Sqrt(F[I])) do
        if (F[I] mod J = 0) then Prime := False;
      if Prime then Inc(PNum);
    end;
  end;
  Writeln(F[I]);
end.
```

```
{2.7}
program Two7T96;
{ -- This program sorts phone bills by zip code and phone #. }
  var
    P, Z, PZ: Array[1..8] of LongInt;
    X:        LongInt;
    N, I, J:  Integer;

begin
  N := 0;
  repeat
    Inc(N);
    Write ('Enter phone #, zip: ');  Readln (P[N], Z[N]);
    PZ[N] := Z[N] * 10000 + P[N];
  until (P[N] = 0) and (Z[N] = 0);
  Dec(N);
  for I := 1 to N - 1 do
    for J := I + 1 to N do
      if PZ[I] > PZ[J] then begin
        X := PZ[I];  PZ[I] := PZ[J];  PZ[J] := X;
        X := P[I];  P[I] := P[J];  P[J] := X;
        X := Z[I];  Z[I] := Z[J];  Z[J] := X;
      end;
  for I := 1 to N do  Writeln (P[I]);
end.
```

```
{2.8}
program Two8T96;
{ -- This program will display number of runs of letters. }
  var
    Let:           String[80];
    Ch:            Char;
    I, H1, H2:     Integer;
    Half1, Half2: Boolean;

begin
  Write ('Enter letters: ');  Readln (Let);
  Half1 := False;  Half2 := False;
  for I := 1 to Length(Let) do begin
    Ch := Let[I];
    if Pos(Ch, 'ABCDEFGHIJKLM') > 0 then begin
      if Half2 then begin
        Inc(H2);  Half2 := False;
      end;
      Half1 := True;
      end
    else begin
      if Half1 then begin
        Inc(H1);  Half1 := False;
      end;
      Half2 := True;
    end;
  end;
  if Half1 then Inc(H1);
  if Half2 then Inc(H2);
  Writeln ('RUNS IN 1ST HALF = ', H1);
  Writeln ('RUNS IN 2ND HALF = ', H2);
end.
```

```
{2.9}
program Two9T96;
{ -- This program reverses the order of letters in each word. }
  var
    S:       String[80];
    Md:      Char;
    I, J, L: Integer;
    W:       String[20];
    Pal:     Boolean;

begin
  Write ('Enter string: ');  Readln (S);  S := S + ' ';
  for I := 1 to Length(S) do begin
    Md := S[I];
    if Md = ' ' then begin
      L := Length(W);  Pal := True;
      for J := 1 to L div 2 do
        if Copy(W, J, 1) <> Copy (W, L-J+1, 1) then Pal := False;
      if Pal then
        for J := 1 to Length(W) do  Write('?')
      else
        for J := L downto 1 do  Write(Copy(W,J,1));
      Write (' ');  W := '';
      end
    else
      W := W + Md;
  end;
  Writeln;
end.
```

```
{2.10}
program Two10T96;
{ -- This program determines day of week for a given date. }
  const
    MonNum: Array[1..12] of Byte = (1,4,4,0,2,5,0,3,6,1,4,6);
    D:      Array[1..7]  of String[9] = ('SATURDAY',
            'SUNDAY', 'MONDAY', 'TUESDAY', 'WEDNESDAY',
            'THURSDAY', 'FRIDAY');
  var
    Month, Day, Year, Last2, Sum, R: Integer;
    LeapYear: Boolean;

begin
  Write ('Enter month, day, year: ');  Readln (Month, Day, Year);
  Last2 := Year mod 100;
  Sum := Last2 + (Last2 div 4);
  LeapYear := (Year Mod 4 = 0) and (Year mod 100 > 0);
  LeapYear := LeapYear or (Year mod 400 = 0);
  if (Month < 3) and LeapYear then
    if (Month = 2) then Inc(Sum,3) else  {-- New Month Number }
  else
    Inc(Sum, MonNum[Month]);
  Inc(Sum, Day);
  Case Year of
    1753..1799: Inc(Sum, 4);
    1800..1899: Inc(Sum, 2);
    2000..2099: Inc(Sum, 6);
    2100..2199: Inc(Sum, 4);
  end;
  R := Sum mod 7;
  Writeln (D[R+1]);
end.
```

```
{3.1}
program Thr1T96;
{ -- This program displays the appearance of 3-dimensional book.}
uses Crt;
  const
    Spaces:        String[16] = '                ';
  var
    T1, T2:        String[17];
    Max, Dif, Row: Byte;

begin
  Write ('Enter title 1: ');  Readln (T1);
  Write ('Enter title 2: ');  Readln (T2);
  if Length(T1) > Length(T2) then
    begin
      Max := Length(T1);  Dif := (Max - Length(T2)) div 2;
      T2 := Copy(Spaces, 1, Dif) + T2 + Copy(Spaces, 1, Dif + 1);
    end
  else
    begin
      Max := Length(T2);  Dif := (Max - Length(T1)) div 2;
      T1 := Copy(Spaces, 1, Dif) + T1 + Copy(Spaces, 1, Dif + 1);
    end;
  ClrScr;
  Writeln ('    /---/!');
  Writeln ('   /   / !');
  Writeln ('  /   /  !');
  Writeln (' /   /   !');
  Writeln ('!---!    !');
  for Row := 1 to Max do begin
    Write ('!');
    Write (Copy (T2, Row, 1), ' ');
    Write (Copy (T1, Row, 1), '!');
    if Row < Max - 3 then
      Writeln (' ':4, '!')
    else
      Writeln (' ': Max - Row + 1, '/');
  end;
  Writeln ('!---!/');
end.
```

```
{3.2}
program Thr2T96;
{ -- This program produces a prime factors tree. }
uses Crt;
  var
    P:                        Array[1..100] of Integer;
    Num, Left, Right, Row, Pr, Dividend, L, R: Integer;

begin
  Write ('Enter number: ');  Readln (NUM);
  ClrScr;  Row := 1;  Writeln (' ':5, Num);
  {-- Position of / and \, determine length of Num }
  Left := 5;  Right := Left + Trunc(Ln(Num) / Ln(10)) + 2;
  repeat
    { -- Find smallest prime that divides number }
    if Num mod 2 = 0 then
      Pr := 2
    else begin
      Pr := 1;
      repeat
        Inc(Pr, 2);
      until (Num mod Pr = 0);
    end;
    Dividend := Num div Pr;
    if Dividend > 1 then begin
      Inc(Row);
      GotoXY (Left, Row);   Write ('/');
      GotoXY (Right, Row);  Writeln ('\');
      L := Trunc(Ln(Pr) / Ln(10));
      R := Trunc(Ln(Dividend) / Ln(10));
      Inc(Row);
      GotoXY (Left - L - 1, Row);  Write (Pr);
      GotoXY (Right + 1, Row);     Writeln (Dividend);
      Left := Right;  Right := Right + R + 2;
    end;
    Num := Dividend;
  until Num = 1;
end.
```

```
{3.3}
program Thr3T96;
{ -- This program simulates a "base four" calculator. }
  var
    Num:                 Array[1..10] of String[6];
    Sym:                 Array[1..10] of Char;
    Ch:                  Char;
    N:                   String[6];
    E:                   String[40];
    I, J, K, L, Dig, X: Byte;
    B10, Total, Pow:     LongInt;

begin
  Write ('Enter base 4 expression: ');  Readln (E);
  E := E + '+';  Sym[1] := '+';
  for I := 1 to Length(E) do begin
    Ch := E[I];
    if (Ch = '+') or (Ch = '-') then
      begin
        Inc(J);  Num[J] := N;  Sym[J+1] := Ch;  N := '';
      end
    else
      N := N + Ch;
  end;
  { -- Convert base 4 numbers to base 10 and perform arithmetic }
  for I := 1 to J do begin
    L := Length(Num[I]);  B10 := 0;
    for J := 1 to L do begin
      Dig := Ord(Num[I,J]) - Ord('0');
      Pow := 1;
      for K := 1 to (L - J) do Pow := Pow * 4;
      B10 := B10 + Dig * Pow;
    end;
    if (Sym[I] = '-') then B10 := (-B10);
    Inc(Total, B10);
  end;
  { -- Convert base 10 number to base 4 }
  if Total < 0 then begin
    Write ('-');  Total := (-Total);
  end;
  J := Trunc(Ln(Total) / Ln(4) + 0.001);
  for I := J downto 0 do begin
    Pow := 1;
    for K := 1 to I do Pow := Pow * 4;
    X := Total div Pow;
    Write (X);
    Total := Total - X * Pow;
  end;
  Writeln;
end.
```

```
{3.4}
program Thr4T96;
{ -- This program calculates contractor's pay=time * rate. }
  var
    Rate, Time: Real;
    St, Fi:     String[7];
    FiHour, StHour, StMin, FiMin, Code: Integer;

begin
  Write ('Enter pay/hour: ');     Readln (Rate);
  Write ('Enter start time: ');   Readln (St);
  Write ('Enter finish time: ');  Readln (Fi);
  Val(Copy(St,1,2), StHour, Code);
  Val(Copy(Fi,1,2), FiHour, Code);
  Val(Copy(St,4,2), StMin,  Code);
  Val(Copy(Fi,4,2), FiMin,  Code);
  { -- Adjust for 12AM and times from 1PM - 11PM }
  if StHour = 12 then
    if Copy(St, 6, 2) = 'AM' then Dec(StHour, 12) else
  else
    if Copy(St, 6, 2) = 'PM' then Inc(StHour, 12);

  if FiHour = 12 then
    if Copy(Fi, 6, 2) = 'AM' then Dec(FiHour, 12) else
  else
    if Copy(Fi, 6, 2) = 'PM' then Inc(FiHour, 12);
  {-- Adjust for a late starting time and early morning finish.}
  if StHour > FiHour then Inc(FiHour, 24);
  {-- Compute difference in time (finish - start) }
  Time := (FiHour - StHour) + (FiMin - StMin) / 60;
  {-- If more than half of time is outside normal hours (7AM-5PM)
   -- then add a shift differential of 10% to rate. }
  if ((7 - StHour) + (0 - StMin) / 60) >= (Time / 2) then
    { -- More than half of time is worked before 7AM }
    Rate := Rate * 1.1;
  if ((FiHour - 17) + (FiMin) / 60) >= (Time / 2) then
    { -- More than half of time is worked after 5PM }
    Rate := Rate * 1.1;
  Writeln ('$', Time * Rate: 6:2);
end.
```

```
{3.5}
program Thr5T96;
{ -- This program displays the button that leads to the others. }
  var
    I, J, K, L, R, C, Press:   Byte;
    N:    Array[1..4, 1..4] of Byte;
    D:    Array[1..4, 1..4] of Char;
    A:    Array[1..4, 1..4] of Boolean;
    Row:  String[12];
    Code: Integer;
    Good: Boolean;

begin
  for I := 1 to 4 do begin
    Write ('Enter row: ');  Readln (Row);
    for J := 1 to 4 do begin
      Val(Row[J*3-2], N[I,J], Code);
      D[I,J] := Row[J*3-1];
    end;
  end;
  for I := 1 to 4 do
    for J := 1 to 4 do begin
      for K := 1 to 4 do
        for L := 1 to 4 do  A[K, L] := False;
      R := I;  C := J;  A[R, C] := True;
      Press := 1;  Good := True;
      repeat
        Case D[R,C] of
          'D': Inc(R, N[R,C]);
          'U': Dec(R, N[R,C]);
          'L': Dec(C, N[R,C]);
          'R': Inc(C, N[R,C]);
        end;
        if A[R, C] then
          Good := False
        else begin
          A[R,C] := True;  Inc(Press);
        end;
      until (not Good) or (Press = 16);
      if Press = 16 then begin
        Writeln ('FIRST BUTTON = ', N[I,J], D[I,J]);
        Writeln ('AT ROW = ', I, ', COL = ', J);
        Exit
      end;
    end;  { -- for J }
end.
```

```
{3.6}
program Thr6T96;
{ -- This program will generate odd size magic squares. }
  var
    N, First, Incr, X, Y, I, J, MagicNum: Integer;
    A:             Array[1..13, 1..13] of Integer;

begin
  Write ('Enter order, first number, increment: ');
  Readln (N, First, Incr);
  X := 1;  Y := (N + 1) div 2;  A[X,Y] := First;
  for I := 2 to N * N do begin
    Dec(X);  Inc(Y);
    if X = 0 then X := N;
    if Y > N then Y := 1;
    if A[X,Y] = 0 then
      A[X,Y] := First + Incr * (I - 1)
    else begin
      Inc(X,2);  Dec(Y);
      if X > N then Dec(X, N);
      if Y = 0 then Y := N;
      A[X,Y] := First + Incr * (I - 1);
    end;
  end;
  { -- Display Magic Number and Square }
  MagicNum := 0;
  for I := 1 to N do  Inc(MagicNum, A[I,1]);
  Writeln ('MAGIC NUMBER = ', MagicNum);
  for I := 1 to N do begin
    for J := 1 to N do
      Write (A[I,J]: 4);
    Writeln;
  end;
end.
```

```
{3.7}
program Thr7T96;
{ -- This program will generate 6x6 magic squares. }
  const
    R: Array[1..4] of Byte = (0, 1, 0, 1);
    C: Array[1..4] of Byte = (0, 1, 1, 0);
  var
    N, First, Incr, X, Y, I, J: Integer;
    FirstN, MagicNum, Sq, Temp: Integer;
    A:             Array[1..3, 1..3] of Integer;
    B:             Array[1..6, 1..6] of Integer;

procedure Generate3x3;
{ -- Generate a 3x3 magic square in A[1..3,1..3] }
begin
  for I := 1 to 3 do
    for J := 1 to 3 do A[I,J] := 0;
  N := 3;
  X := 1;  Y := (N + 1) div 2;  A[X,Y] := First;
  for I := 2 to N * N do begin
    Dec(X);  Inc(Y);
    if X = 0 then X := N;
    if Y > N then Y := 1;
    if A[X,Y] = 0 then
      A[X,Y] := First + Incr * (I - 1)
    else begin
      Inc(X,2);  Dec(Y);
      if X > N then Dec(X, N);
      if Y = 0 then Y := N;
      A[X,Y] := First + Incr * (I - 1);
    end;
  end;
end;

begin
  Write ('Enter first number, increment: ');
  Readln (FirstN, Incr);
  { -- Four 3x3 squares are made for the 6x6 matrix B[]
    -- upper-left, bottom-right, upper-right, bottom-left. }
  for Sq := 0 to 3 do begin
    First := FirstN + Sq * 9 * Incr;
    Generate3x3;
    for I := 1 to 3 do
      for J := 1 to 3 do
        B[R[Sq+1] * 3 + I, C[Sq+1] * 3 + J] := A[I,J];
  end;
  { -- Transpose three cells }
  Temp := B[1,1];  B[1,1] := B[4,1];  B[4,1] := Temp;
  Temp := B[2,2];  B[2,2] := B[5,2];  B[5,2] := Temp;
  Temp := B[3,1];  B[3,1] := B[6,1];  B[6,1] := Temp;
  { -- Display Magic Number and 6x6 matrix }
  MagicNum := 0;
  for I := 1 to 6 do  Inc(MagicNum, B[I,1]);
  Writeln ('MAGIC NUMBER = ', MagicNum);
  for I := 1 to 6 do begin
```

```
    for J := 1 to 6 do
      Write (B[I,J]: 4);
    Writeln;
  end;
end.
```

```
{3.8}
program Thr8T96;
{ -- 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.
```

```
{3.9}
program Thr9T96;
{ -- This program produces a precedence of jobs to run. }
  var
    Num, I, J, K, L, DepLeft, UNum, P, St: Byte;
    Job:                  String[3];
    Dep:                  String[6];
    U, U2, Jobs, NewU2: String[24];
    A, B:                 Array[1..8] of String[3];
    Marked:               Array[1..8] of Boolean;
    NoJob, ValidJob:    Boolean;


begin
  Write ('Enter number of dependencies: ');  Readln (Num);
  U := '';
  for I := 1 to Num do begin
    Write ('Enter dependency: ');  Readln (Dep);
    Dep := Dep + ' ';
    A[I] := Copy(Dep, 1, 3);
    B[I] := Copy(Dep, 4, 3);
    { -- Store unique jobs in string }
    if Pos(A[I], U) = 0 then U := U + A[I];
    if Pos(B[I], U) = 0 then U := U + B[I];
  end;
 { -- Since there is a unique order for all the jobs,
   -- every job will have its successor somewhere in B[].
   -- 1) search all B[] for the only job missing.
   -- 2) exclude all dependencies with this job in it.
   -- 3) search all B[] for the next only job missing.
   -- 4) repeat steps 2 and 3 until the final dependency is left.}
  L := Length(U);  UNum := L div 3;  U2 := U;
  DepLeft := Num;  Jobs := '';
  while DepLeft > 1 do begin
    for I := 1 to Num do  Marked[I] := False;
    for I := 1 to Num do begin
      P := Pos(B[I], U2);
      if P > 0 then Marked[ (P+2) div 3 ] := True;
    end;
    NoJob := True;  I := 0;
    while NoJob and (I < UNum) do begin
      Inc(I);  St := I * 3 - 2;
      Job := Copy(U2, St, 3);
      ValidJob := (Pos(Job, Jobs) = 0) and (Job <> '   ');
      if ValidJob and not Marked[I] then begin
        Jobs := Jobs + Job;
        for K := 1 to Num do
          if A[K] = Job then begin
            A[K] := '*';  B[K] := '*';
            Dec(DepLeft);
          end;
        NewU2 := Copy(U2, 1, St-1) + '   ';
        U2 := NewU2 + Copy(U2, St + 3, L - St - 2);
        NoJob := False;
      end;
    end;  { -- while }
  end;  { -- while }
  { -- Last dependency is concatenated }
  for I := 1 to Num do
    if A[I] <> '*' then Jobs := Jobs + A[I] + B[I];
  Writeln ('JOBS MUST BE RUN IN THIS ORDER: ', Jobs);
end.
```

```
{3.10}
program Thr10T96;
{ -- This program finds a perfect square with digits 1-9. }
  var
    A, N, Num, Min, NumMin, NumMin2: LongInt;
    I, B, Z, L, Code: Integer;
    Digits:           String[9];
    Good:             Boolean;
    Count:            Byte;

procedure CheckDigits;
{ -- Determine number of swaps made and store in count }
var
  D:          Array[1..9] of Byte;
  I, J, Temp: Byte;

begin
  for I := 1 to 9 do  Val(Digits[I], D[I], Code);
  Count := 0;
  for I := 1 to 9 do
    if D[I] <> I then begin
      J := I + 1;
      While (J < 9) and (D[J] <> I) do  Inc(J);
      Temp := D[I];  D[I] := D[J];  D[J] := Temp;
      Inc(Count);
    end;
end;

{ -- Main program }
begin
  Min := 9;
  for Num := 10001 to Trunc(Sqrt(987654321)) do begin
    A := Num * Num;
    Str(A, Digits);
    Good := True;  L := 1;
    while (L <= 9) and Good do begin
      if Pos(Chr(48+L), Digits) = 0 then Good := False;
      Inc(L);
    end;
    if Good then begin  {-- Found perfect square w/unique digits}
      CheckDigits;
      if Count < Min then begin
        Min := Count;  NumMin := A;  NumMin2 := Num;
      end;
    end;
  end;
  { -- Display the perfect square needing least num of swaps. }
  Str(NumMin, Digits);
  Writeln (Digits, ' IS THE SQUARE OF ', NumMin2);
  Write   ('AND WAS FORMED BY EXCHANGING ', Min);
  Writeln (' PAIRS OF DIGITS');
end.
```