FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '89

1.1 Write a program to print the phrase 1989 COMPUTER CONTEST on each line of the screen such that each successive phrase is indented one extra space. Example:

```
    OUTPUT:  1989 COMPUTER CONTEST
              1989 COMPUTER CONTEST
               1989 COMPUTER CONTEST
                :
                 :
                  :
```

1.2 A database is a collection of data, or information representing abstract entities. Databases require much storage space. Most businesses measure their databases in terms of gigabytes of data. A gigabyte is equivalent to 1024 or 2^10 megabytes (a megabyte is approximately a million bytes). Write a program to represent a gigabyte value (a gigabyte is approximately a billion bytes) in its equivalent number of megabytes. Input will consist of a positive integer less than 30. Example:

```
     INPUT: Enter number of gigabytes: 14
    OUTPUT: 14336 MEGABYTES
```

1.3 Write a program to display a word in a backward-L format. The given word is to be displayed vertically and horizontally so that they share the same last letter. Example:

```
     INPUT: Enter word: EXAMPLE

    OUTPUT:       E
                  X
                  A
                  M
                  P
                  L
            EXAMPLE
```

1.4 Write a program to produce the following pattern for an input integer, N, with N between 1 and 9 inclusive:

```
          1              Example:    INPUT: Enter N: 4
         2 2                        OUTPUT:     1
        3   3                                  2 2
       .     .                                3   3
      .       .                              4     4
     N         N
```

1.5 Anno Domini is Latin for "in the year of our Lord" and is abbreviated as A.D.. In 525 A.D., Pope John I asked the monk Dionysius Exiguus to begin a Christian system of dating events, starting with the year Dionysius believed Christ was born. The years before the birth of Christ are termed B.C., while the years after the birth of Christ are termed A.D. The following is the order of years: ... 2 B.C., 1 B.C., 1 A.D., 2 A.D., ... with Christ's birth being in the year 1 A.D. Today, we know that the monk was in error. Even though we continue to use his dating system, biblical scholars currently believe that Christ was born four years earlier than what the monk thought. Write a program that corrects modern dates to account for this change. Examples:

```
 INPUT: Enter date: 4
        Enter A.D. or B.C.: B.C.
OUTPUT: 1 A.D.

 INPUT: Enter date: 1
        Enter A.D. or B.C.: A.D.
OUTPUT: 5 A.D.

 INPUT: Enter date: 5
        Enter A.D. or B.C: B.C.
OUTPUT: 1 B.C.
```

1.6 Many computer systems require a user to enter a password to ensure that the appropriate person is accessing his/her files of information. Write a program to prompt a user for a password with the words "ENTER PASSWORD: ". For this program the user's password is ITSME. The user has up to 3 chances to enter the correct password. If it is correct then display the message "YOU HAVE ACCESS". For the first two tries, if an incorrect password is entered, display "INVALID PASSWORD:" and prompt for another password. After 3 incorrect entries, display "YOU ARE TRESPASSING" and exit. Example:

```
OUTPUT/INPUT:  ENTER PASSWORD: TRY1
OUTPUT/INPUT:  INVALID PASSWORD: TRY2
OUTPUT/INPUT:  INVALID PASSWORD: TRY3
OUTPUT:        YOU ARE TRESPASSING

OUTPUT/INPUT:  ENTER PASSWORD: TRYAGAIN
OUTPUT/INPUT:  INVALID PASSWORD: ITSME
OUTPUT:        YOU HAVE ACCESS
```

1.7 Write a program to determine the "best" Data Base Management System (DBMS) to use. A DBMS is a set of programs that access a collection of interrelated data, called a database. A DBMS is "good" if it provides an environment that is both convenient and efficient to use in retrieving data from the database and in storing data into the database.

First, N will be input as the number of DBMS to consider. Next, N names pertaining to the DBMS will be entered, each followed by it's respective convenience rank and efficiency rank (both between 1 and 10 inclusive). The "best" DBMS is determined by the highest total rank for convenience and efficiency. Display the name of the best DBMS that "IS BEST". In the example below, Amy is best because (3+9) is larger than (10+1) which is larger than (3+4). Example:

```
     INPUT:  Enter N: 3
             Enter DBMS name: DOUG
             Enter convenience, efficiency: 3, 4
             Enter DBMS name: AMY
             Enter convenience, efficiency: 3, 9
             Enter DBMS name: CRAIG
             Enter convenience, efficiency: 10, 1
    OUTPUT:  AMY IS BEST
```

1.8 Write a program to display the elements of a list of integers, without repetition, in the order of their appearance in the list, separated by one space. One number at a time will be input. Termination of the list will be designated by the input of -999. Example:

```
     INPUT: Enter #: 2
            Enter #: 3
            Enter #: -1
            Enter #: 3
            Enter #: 2
            Enter #: -5
            Enter #: -999

    OUTPUT: 2 3 -1 -5
```

1.9 Often statisticians compute such large probabilities that they are difficult for the average person to comprehend. To help illustrate such a number, a real life model is used. Write a program to illustrate the probability of "1 out of N", where N is a large real number given in scientific "E" notation. Output will consist of the nearest integer of FEET DEEP of silver dollars that the state of Texas needs to be covered to be equivalent to the number N. Output will be less than 1000.

Assume that Texas has 262,134 square miles of land, while a silver dollar is 1 1/2 inches in diameter and 3/32 inch in thickness. Assume that the silver dollars will be piled in rows and columns. Examples:

```
                                          | silver dollars
 INPUT: Enter probability: 1E17           | are piled like:
OUTPUT: 2 FEET DEEP                       | OOOOOOOOOO
                                          | OOOOOOOOOO
 INPUT: Enter probability: 2.6E18         | OOOOOOOOOO
OUTPUT: 43 FEET DEEP                      | OOOOOOOOOO
```

1.10 Memory is a large array of bytes, each with its own address. Each program in a computer system deals with particular logical addresses. The memory mapping hardware converts logical addresses into physical addresses. Logical addresses are in the range of 0 to Max. The corresponding Physical addresses are in the range of R+0 to R+Max, where the value R is the lower bound.

Write a program to map a given logical address and segment to the corresponding physical address. Each segment starts at a specific physical address (base) and has a given length. The physical address can be determined using the data in the table below by adding the base (smallest physical address in a specified segment) to the given logical address. If the logical address specified is greater than the length of the segment, display ADDRESSING ERROR. Input will be the segment number followed by the logical address to convert. Output will be an error message (ADDRESSING ERROR) or the physical address. Repeat input until a segment number greater than 4 is entered.

Data:

Segment	Base	Length
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Example:

```
 INPUT: Enter Seg#, Address: 1, 11
OUTPUT: 2311
 INPUT: Enter Seg#, Address: 3, 600
OUTPUT: ADDRESSING ERROR
 INPUT: Enter Seg#, Address: 2, 95
OUTPUT: 185
 INPUT: Enter Seg#, Address: 5, 0
OUTPUT: (program terminates)
```

2.1 Write a program to display the value of F(x), given x as an input positive integer between 1 and 10 inclusive, and given:

$$F(1) = F(2) = F(3) = 1, \text{ and}$$

$$F(x+1) = \frac{F(x)*F(x-1)+2}{F(x-2)} \quad \text{for x greater than 3}$$

Output must be of the form "F(x)=" F(x), where the first x is substituted by the input value of x, and the second F(x) is the actual value of the function. Examples:

```
 INPUT: Enter x: 4              INPUT: Enter x: 6
OUTPUT: F(4)= 3                OUTPUT: F(6)= 17
```

2.2 Write a program to print out the prime factorization equation for a given positive integer. The prime numbers must be in increasing order and separated by an "X". Examples:

```
 INPUT: Enter #: 90              INPUT: Enter #: 17
OUTPUT: 2 X 3 X 3 X 5           OUTPUT: 17
```

2.3 Write a program to display a word without its vowels: a,e,i,o,u, and exclude y. Example:

```
 INPUT: Enter word: CONTEST
OUTPUT: CNTST


 INPUT: Enter word: ANSWER
OUTPUT: NSWR
```

2.4 In order to write a program, a programmer must choose appropriate names for variables, constants, procedures, etc. Each identifier should have a name that correctly identifies its purpose and function in the program. However, if a name is too long, then it is burdensome to write and read in a program and may occupy more space on a line than is necessary. Good short names that properly describe an object's function are better than long names.

For the sake of brevity, write a program to produce the shortest possible names for a set of 6 identifiers in a program so that each one is distinguishable using the following method. If two or more identifiers start with the same character(s) compare each identifier character by character until they are distinguishable. Truncate the remainder of the identifier after the distinguishable character (see MINIMUM, MAXIMUM, and MAXNUM in example). Assume that the language can distinguish identifiers with any amount of letters, if they differ. Example:

```
     INPUT: Enter name: AVERAGE          OUTPUT: A
            Enter name: MINIMUM                  MI
            Enter name: MAXIMUM                  MAX
            Enter name: MAXNUM                   MAXN
            Enter name: POSITION                 POS
            Enter name: POINTER                  POI
```

2.5 Write a program to display the number of distinguishable permutations of letters of a given word. The mathematical formula for calculating such a number is given as the factorial, (!), of the number of letters in the word divided by the product of factorials of the number of times a letter appears. Examples:

```
     INPUT: Enter word: WITTICISM
    OUTPUT: 30240
```

(there are 9 letters, 3 I's, and 2 T's, 9! / (3! x 2!) = 30240)

```
     INPUT: Enter word: ELEMENT
    OUTPUT: 840
```

(there are 7 letters, 3 E's, 7! / 3! = 840)

NOTE: A factorial is the product of all the integers from 1 to the number. (e.g. 5!=5x4x3x2x1 = 120) Since 1! = 1, letters that appear only once are not relevant.

2.6 Write a program to simulate a word processor that underlines text between two asterisks. Underlining mode begins at the first asterisk and ends at the next asterisk. Many different segments of a line may be underlined. The line of text will contain no more than 40 characters. The program is to accept a line of text, clear the screen, display the line, skip a line, display the line without it's embedded asterisks, then underline (with hyphens on the next line) the words between the first and second asterisks, the third and fourth, and so on. Example:

```
 INPUT: I *REALLY* THINK THAT *WE WILL WIN*

OUTPUT: (Screen is cleared)
        I *REALLY* THINK THAT *WE WILL WIN*

        I REALLY THINK THAT WE WILL WIN
          ------            -----------
```

2.7 Write a program to compute an integer expression involving two positive integers separated by one of the following: +, -, *, or /. Each integer will contain no more than 4 digits and the result is guaranteed to be an integer between -30,000 and 30,000. Examples:

```
 INPUT: 80/5        INPUT: 543*21        INPUT: 999-5556
OUTPUT: 16         OUTPUT: 11403        OUTPUT: -4557
```

2.8 Game theory is a mathematical theory formally dealing with competitive situations. Emphasis is placed on the decision-making processes of the adversaries. In a two-person game, a player may use a two dimensional matrix of numbers to represent a payoff table. Each number represents an amount of win or loss. Each player tries to minimize his maximum losses. If there exists an element in the table that is both the minimum in its row and the maximum in its column, then it is called a saddle point. When a saddle point exists, neither player has an advantage over his opponent.

Write a program to determine the saddle point element (each set of data will have a saddle point) and its row position and column position. The program is to first accept the number of rows and columns in the table. Then, the program accepts the elements of the table starting with each column element in row 1, then each column element of row 2, etc. The output must be of the form, SADDLE POINT = # AT ROW # COL #. Example:

```
 INPUT: Enter # Rows, # Cols: 3, 3
        Enter Row1 Col1: -3
        Enter Row1 Col2: -2
        Enter Row1 Col3: 6
        Enter Row2 Col1: 2
        Enter Row2 Col2: 0
        Enter Row2 Col3: 2
        Enter Row3 Col1: 5
        Enter Row3 Col2: -2
        Enter Row3 Col3: -4

OUTPUT: SADDLE POINT = 0 AT ROW 2 COL 2
```

2.9 Write a program to sort a set of dates entered. First, accept the number of dates to sort. Next, accept each date by first accepting the entire name of the month, then the day, then the year as integers. Display the dates in order in the form: MONTH DAY YEAR. Example:

```
 INPUT: Enter # of dates: 3
        Enter month: JANUARY
        Enter day:   1
        Enter year:  1978

        Enter month: FEBRUARY
        Enter day:   20
        Enter year:  1977

        Enter month: JANUARY
        Enter day:   27
        Enter year:  1978

OUTPUT: FEBRUARY 20 1977
        JANUARY 1 1978
        JANUARY 27 1978
```

2.10 Because Ms. Heindel is such a nice music teacher, she is allowing her students to retake quiz 4, since her class did rather poorly. Write a program to display the table of names and grades (with column headings) as shown below, then accept the 5 new grades for the students for quiz 4, in their order of listing. Then clear the screen and display the final report with averages for each person and for each quiz and for the overall class. Averages must be accurate to 2 decimal places and be aligned in the proper column. The headings in the final report must be centered, as shown below. With the exception of spacing, the content must look as follows:

```
  RUN PROGRAM:

       OUTPUT:    NAME      Q1     Q2     Q3     Q4

               D. WOOLY    100     92     90     90
               M. SMITH     55     75     70     65
               C. BROWN     94     70     62     70
               R. GREEN     90     74     80     85
               T. STONE     85     98    100     70

       INPUT: Enter 5 grades for quiz 4: 98, 70, 75, 90, 80

      OUTPUT: (Screen is cleared)
                          MS. HEINDEL'S MUSIC CLASS
                                 FINAL GRADES
                                 SPRING 1989

                 NAME       Q1     Q2     Q3     Q4  AVERAGE

               D. WOOLY    100     92     90     98   95.00
               M. SMITH     55     75     70     70   67.50
               C. BROWN     94     70     62     75   75.25
               R. GREEN     90     74     80     90   83.50
               T. STONE     85     98    100     80   90.75

               AVERAGE:  84.80  81.80  80.40  82.60

               CLASS AVERAGE: 82.40
```

3.1 Write a program to simulate a mini spell checker. Given a word, determine if it is CORRECT or MISSPELLED. A word is misspelled if it violates any of the following spelling rules:

- 'E' does not appear before the suffixes 'ING', 'IBLE', 'ABLE'
- 'I' before 'E' except after 'C'
- A letter may appear no more than twice consecutively

Examples:

```
INPUT: Enter word: APPPLE     OUTPUT: MISSPELLED
INPUT: Enter word: PIE        OUTPUT: CORRECT
INPUT: Enter word: EATING     OUTPUT: CORRECT
INPUT: Enter word: NOTEABLE   OUTPUT: MISSPELLED
INPUT: Enter word: RECIEVE    OUTPUT: MISSPELLED
```

3.2 Write a program to calculate the positive amount of (V)olume for a given (P)ressure according to the following thermo-dynamics equation:

P*V*V*V*14.14 - P*V*9062.599 - 23511.9*V*V + 988686.1*V =400943.0

The program will first simulate the values of V for the following values of P: 0.05, 0.70, 10.00, 70.00. Next, a positive value of P (less than 100) is entered and the value of V is computed and displayed. Values of V must be rounded to the nearest ten-thousandth (0.0001). Display both the value of P and V, as shown below, where ? represents the computed value for V. Example:

```
RUN PROGRAM:

OUTPUT: P =  0.05  V = 0.4097
        P =  0.70  V = ?
        P = 10.00  V = ?
        P = 70.00  V = 1.2263
 INPUT: Enter value for P: 30.00
OUTPUT: P = 30.00  V = 0.5699
```

3.3 Write a program to magnify an input positive integer. Each digit must be displayed in block format (made of *) with dimensions determined by its magnification (1, 2, or 3). At most 4 digits will be input for magnification of 1; At most 3 digits for magnification of 2, and at most 2 digits for magnification of 3. The dimensions of a block number are as follows:

Magnification	Dimensions	Space Between Digits
1	5 rows by 4 columns	(2 space separation)
2	9 rows by 8 columns	(4 space separation)
3	13 rows by 12 columns	(6 space separation)

Examples:

```
 INPUT: Enter number: 3124
        Enter magnification: 1

OUTPUT: ****     *  ****  *  *        (note: digit 1 is in
           *     *     *  *  *         right most column of
        ****     *  ****  ****         block format)
           *     *  *        *
        ****     *  ****     *
```

```
 INPUT: Enter number: 567
        Enter magnification: 2

OUTPUT: ********    ********    ********
        *           *                  *
        *           *                  *
        *           *                  *
        ********    ********           *
               *    *      *           *
               *    *      *           *
               *    *      *           *
        ********    ********           *
```

```
 INPUT: Enter number: 89
        Enter magnification: 3

OUTPUT: ************      ************
        *          *      *          *
        *          *      *          *
        *          *      *          *
        *          *      *          *
        *          *      *          *
        ************      ************
        *          *                 *
        *          *                 *
        *          *                 *
        *          *                 *
        *          *                 *
        ************                 *
```

3.4 Write a program to produce a calendar for an input month of a year. The month will be any number between 1 and 12 inclusive, and the year will be between 1901 and 1999 inclusive. Every year (in this century) divisible by 4 is a leap year. January 1, 1901 was a Tuesday. Each heading of the month and year will be centered above the calendar. The rest of the calendar should appear exactly as shown below. Sunday's column has a 2 space margin; all the other days have a 4 space margin in which the numbers are right justified. Examples:

```
 INPUT: Enter month, year: 1, 1901
OUTPUT:         JANUARY 1901
           S   M   T   W   T   F   S
          --------------------------
                   1   2   3   4   5
           6   7   8   9  10  11  12
          13  14  15  16  17  18  19
          20  21  22  23  24  25  26
          27  28  29  30  31

 INPUT: Enter month, year: 2, 1988
OUTPUT:        FEBRUARY 1988
           S   M   T   W   T   F   S
          --------------------------
               1   2   3   4   5   6
           7   8   9  10  11  12  13
          14  15  16  17  18  19  20
          21  22  23  24  25  26  27
          28  29
```

3.5 Write a program to determine all the possible ways to position 5 queens on a 5x5 chess board so that none of them can attack another. A queen can attack another queen if the other queen is in the same row or column or diagonal. For each solution, display the column the queen is in pertaining to the row. Output display must be of the format shown below, with each column number in row 1 non-decreasing for each solution. If 2 solutions have the same column number in row 1, put the solution with the smallest column number in row 2 first.

```
OUTPUT:  ROWS =  1 2 3 4 5
         -----------------
         COLUMNS
                 1 3 5 2 4
                 : : : : :
                 : : : : :
                 5 3 1 4 2
```

The first solution has queens in (row 1, col 1), (row 2, col 3), (row 3, col 5), (row 4, col 2), (row 5, col 4). The last solution has queens in (row 1, col 5), (row 2, col 3), (row 3, col 1), (row 4, col 4), (row 5, col 2).

3.6 Write a program to display the product of two large integers in a given base. Integers will be at most 30 digits in length, and the base will be between 2 and 10 inclusive (both the input integers and output integers will be in the given base). Examples:

```
 INPUT: Enter base: 8
        Enter first integer: -12345670123456701234567
        Enter second integer: 7654321076543210
OUTPUT: -1217053361466167165730670440233335104070

 INPUT: Enter base: 10
        Enter first integer:  1234567890
        Enter second integer: 9999999999
OUTPUT: 12345678898765432110
```

3.7 Write a program to determine the most efficient way to represent change. Input will consist of the COST, the given AMOUNT, and the COIN that is unavailable for making change. Pennies, nickels, dimes, and quarters are available for making change, with the exception of the COIN input as missing. The most efficient change is defined as the combination that requires the fewest number of coins.

Output must display the change returned, starting with the smallest denomination, and not including the missing coin. Next, the total change returned is displayed. Change returned will not exceed 99 cents. If only one coin of a denomination is used, then the singular form of the name must be used (as in PENNY). Otherwise, the plural form is used (as in PENNIES). Examples:

```
 INPUT: Enter cost, amount: 14.41, 15.00
        Enter missing coin: NICKEL
OUTPUT: 4 PENNIES
        3 DIMES
        1 QUARTER
        TOTAL CHANGE RETURNED = 59 CENTS

 INPUT: Enter cost, amount: 4.40, 5.00
        Enter missing coin: DIME
OUTPUT: 0 PENNIES
        2 NICKELS
        2 QUARTERS
        TOTAL CHANGE RETURNED = 60 CENTS
```

3.8 Write a program to find the corner coordinates of rectangles consisting of 1's in a two dimensional table of binary numbers. The table consists of 6 rows by 7 columns. Six numbers in base 10 (each less than 128) are entered for each row. These numbers are converted into base 2 and padded with 0's on the left (if necessary) to fit into the 7 columns of that row. The program will display this table. Next, the computer finds all rectangles filled with 1's and displays the coordinates of the upper left corner and the bottom right corner of each rectangle. A rectangle must have dimensions of at least 2. No rectangles will overlap (i.e., in cases where rectangle is contained in another rectangle, give the coordinates of the larger rectangle). Display each set of coordinates on a separate line with parenthesis and commas, as shown below. Example:

```
INPUT: Enter number: 127        OUTPUT: 1111111
       Enter number: 123                1111011
       Enter number: 23                 0010111
       Enter number: 99                 1100011
       Enter number: 88                 1011000
       Enter number: 57                 0111001

                                        (1,6)(4,7)
                                        (1,1)(2,4)
                                        (5,3)(6,4)
```

(Note: 3 lines of coordinates may appear in any order)

3.9 Write a program to determine the five word combination that gives the greatest point value according to the following rules. Two sets of five words each (shown below in the first output) are given as data. The words in each set are placed on top of each other in such a way that BINGO is spelled in the first column of the first set of words and the second column of the second set of words. Each word has a numerical value as determined by summing the values of each letter as shown below:

```
      A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P
      9 14  1 16 20  5 10  2 21 17  6 25 12  3 22 18

      Q  R  S  T  U  V  W  X  Y  Z
     24  7 13 26 15 11 19  4 23  8
```

The sum of the letters in each word is displayed to the right of the word and a total of all the word sums is placed beneath the five word sums. The set of five words with the greatest total has 3 asterisks placed underneath its total.

The program will start with the 10 given words displaying the numerical sums. Next, the program will accept any number of new five letter words to replace those words in the original two sets. For a new word to replace another word currently in the set, its total must be larger than the total of the word it is replacing (a new word may be used in either or both lists). Also, the entire set of words must still spell BINGO in the first or second columns as it did previously. Pressing the <ENTER> or <RETURN> key will end the input of words and display the new word list and sums (with the larger total sum underlined with three asterisks). The program then accepts more words, or it may quit if the word QUIT is entered. The second set of five words (with BINGO down the second column) must be placed to the right of the first set of words, as shown below. Example:

```
    OUTPUT: BIBLE  94   OBESE  89
            IDYLL 110   TITHE  95
            NOISE  79   INLET  95
            GULLY  98   IGLOO 100
            OBESE  89   TOWER  94
                  470         473
                              ***
     INPUT: Enter word: NOTED
            Enter word: BOOST
            Enter word: OLIVE
            Enter word: ONION
            Enter word: (Enter key pressed)
    OUTPUT: BOOST  97   OBESE  89
            IDYLL 110   TITHE  95
            NOTED  87   INLET  95
            GULLY  98   IGLOO 100
            OLIVE  99   BOOST  97
                  491         476
                  ***
      INPUT: Enter word: QUIT
     OUTPUT: (program terminates)
```

3.10 Write a program to determine the number of distinguishable ways to place and orient a cube with a solid color on each of its six sides. The program will ask for the one letter color symbol for each of the sides in the following order: TOP, FRONT, BOTTOM, BACK, RIGHT, LEFT. Output will be a statement declaring the NUMBER OF DISTINGUISHABLE CUBES--different ways to position and orient the cube. Examples:

```
     INPUT: Enter TOP side:    Y
            Enter FRONT side:  Y
            Enter BOTTOM side: Y
            Enter BACK side:   Y
            Enter RIGHT side:  Y
            Enter LEFT side:   Y
    OUTPUT: NUMBER OF DISTINGUISHABLE CUBES = 1

     INPUT: Enter TOP side:    G
            Enter FRONT side:  G
            Enter BOTTOM side: G
            Enter BACK side:   G
            Enter RIGHT side:  G
            Enter LEFT side:   Y
    OUTPUT: NUMBER OF DISTINGUISHABLE CUBES = 6

     INPUT: Enter TOP side:    B
            Enter FRONT side:  B
            Enter BOTTOM side: O
            Enter BACK side:   Y
            Enter RIGHT side:  G
            Enter LEFT side:   R
    OUTPUT: NUMBER OF DISTINGUISHABLE CUBES = 24
```