FLORIDA HIGH SCHOOLS COMPUTING COMPETITION '88

1.1 Write a program to first clear the screen and then print the phrase:

THE BEST COMPUTER CONTEST!

ten times on the first ten consecutive lines on the screen, each beginning in column 1.

1.2 Write a program to determine if a given number is an INTEGER value or only a REAL value. Example:

```
 INPUT: Enter #: 2.0
OUTPUT: INTEGER

 INPUT: Enter #: -1.5
OUTPUT: REAL
```

1.3 Assume that the surface of a floppy diskette has 40 tracks, and there are 8 sectors per track with 512 bytes/sector. Write a program to determine the number of bytes of information that is on the surface of N floppy diskettes, where N is input as a positive integer. Example:

```
 INPUT: Enter N: 4

OUTPUT: 655360
```

1.4 A complete computer consists of a CPU, PRIMARY memory, SECONDARY memory, an INPUT device, and an OUTPUT device. Given four of the computer's components as input, determine which component is missing. Example:

```
 INPUT: Enter component: SECONDARY
        Enter component: OUTPUT
        Enter component: CPU
        Enter component: INPUT

OUTPUT: PRIMARY
```

1.5 Write a program to outline the perimeter of the screen with asterisks (*) and to divide the screen into four approximately congruent rectangles using *'s. Print the numbers 1, 2, 3, and 4 centered in each rectangle as shown below in miniature.

```
OUTPUT: *************
        *     *     *
        *  1  *  2  *
        *     *     *
        *************
        *     *     *
        *  3  *  4  *
        *     *     *
        *************
```

1.6 Many times a long computer term will be abbreviated by forming an acronym: Electronic Numerical Integrator And Calculator becomes ENIAC. Write a program to make acronyms for given sets of words. Each word in a set will be separated by a space. Acronyms are formed from the first letter of each word in a set. Example:

```
 INPUT: Enter words: RANDOM ACCESS MEMORY
OUTPUT: RAM

 INPUT: Enter words: CENTRAL PROCESSING UNIT
OUTPUT: CPU
```

1.7 Computers are often classified into three categories: MICRO computers, MINI computers, and MAINFRAME computers. MICRO computers are small enough to be placed on a small table and are often used during high school computer programming contests. A MINI computer's CPU and main memory are usually placed on the floor in a case and stand 3 to 5 feet high. MAINFRAME computer systems usually occupy a large room and can be accessed by more than one hundred terminals at the same time. Given the name of a computer and the type of computer, print out a list of the computer names in ascending order according to their size. MAINFRAME is larger than a MINI which is larger than a MICRO. Three computers, each of a different category, will be entered (name, then the type of computer). Example:

```
 INPUT: Enter name: FRED
        Enter type: MINI
        Enter name: WYLBUR
        Enter type: MAINFRAME
        Enter name: GEORGE
        Enter type: MICRO

OUTPUT: GEORGE
        FRED
        WYLBUR
```

1.8 A display in a grocery store will consist of cans stacked on top of each other with the bottom row having N cans in it and each row above it having two cans less than the supporting row. Write a program to find out how many cans will be needed in the display if N is input as the number of cans for the bottom row. N will be greater than 2. Example:

```
 INPUT: Enter N: 5
OUTPUT: 9

 INPUT: Enter N: 6
OUTPUT: 12
```

1.9 A queue is a first-in first-out (FIFO) storage. Objects are removed from a queue in the same order as they are added. A waiting line in a supermarket is an example of a queue. Write a program to place numbers on a queue and then retrieve from or add to the queue. The user is to enter the option to ADD to the queue or TAKE from the queue or QUIT. If the user enters ADD then the program accepts a value to place on the queue. If the user enters TAKE then the first value in the queue is removed and displayed. If QUIT is entered then the program ends. Example:

```
 INPUT: Enter command: ADD
        Enter integer: 5
        Enter command: ADD
        Enter integer: 8
        Enter command: TAKE
OUTPUT: 5
 INPUT: Enter command: ADD
        Enter integer: 2
        Enter command: TAKE
OUTPUT: 8
 INPUT: QUIT
OUTPUT: (program terminates)
```

1.10 Write a program to determine the events of history that occurred between two dates input. Use the following DATA statements in your program (you may store the DATA elements in an array). Each DATA statement contains a year of an invention, person(s) responsible, and invention. Output must display the person followed by the word INVENTED and the invention for each event that is between two years that are input. The first date will precede the second date in time. Example:

```
DATA 1642,"BLAISE PASCAL","ADDING MACHINE"
DATA 1801,"JOSEPH JACQUARD","PUNCHCARD AND WEAVING LOOM"
DATA 1830,"CHARLES BABBAGE","DESIGN OF ANALYTIC ENGINE"
DATA 1890,"HERMAN HOLLERITH","PUNCHCARD TABULATING MACHINE"
DATA 1944,"HOWARD AIKEN","MARK I"
DATA 1946,"ECKERT AND MAUCHLY","ENIAC"
DATA 1949,"VON NEUMAN","EDVAC"

INPUT: Enter years: 1640, 1820

OUTPUT: BLAISE PASCAL INVENTED ADDING MACHINE
        JOSEPH JACQUARD INVENTED PUNCHCARD AND WEAVING LOOM
```

2.1 Write a program to display a solid diamond of asterisks of height and length of N, where N is an odd number greater than 1 and less than 22. The middle row has N asterisks and each consecutive row differs by 2 asterisks. The first and last rows each have one asterisk. Example:

```
 INPUT:  Enter N: 5

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

2.2 Sorting is the arranging of elements of a list into order based on the value of a particular field within the elements. There are several algorithms to accomplish the task of sorting including: bubble sort, shell sort, and quick sort. Each sorting algorithm's efficiency depends upon the number of elements to be sorted and the particular order in which they are originally given. The average number of comparisons needed to sort a list of N elements is given as:

```
N * (N-1) / 2                     for BUBBLE SORT
N * ((Log base 2 of N) squared)   for SHELL SORT
N * (Log base 2 of N)             for QUICK SORT
```

Assuming that the efficiency depends only on the number of elements in the list, N, determine the efficiency order of the different sorting algorithms and print them from most efficient to least efficient. N will be greater than 2. Example:

```
 INPUT: Enter N: 128

OUTPUT: QUICK SORT
        SHELL SORT
        BUBBLE SORT
```

2.3 When a group of no more than 200 was asked to divide itself into groups of 2, 1 person was left over. When asked to divide into groups of 3, 5, 7, there were 2, 1, and 2 left over respectively. Write a program to find out how many people were in this group and display this number.

2.4 Random numbers between 0 and 9999 can be generated according to the following method. Take a 4-digit number as the seed, square it, and pad 0's at the end (if needed) to make the product an 8-digit number. Use the four digits in the middle of the number as the random number and the next seed. Write a program to produce the first 5 random numbers (without leading 0's, if the number is less than four digits) for a given seed of four digits. Example:

```
 INPUT: Enter seed: 3571

OUTPUT: 7520      (because 3571 * 3571 = 12 7520 41 )
        5504      (because 7520 * 7520 = 56 5504 00 )
        2940      (because 5504 * 5504 = 30 2940 16 )
        4360      (because 2940 * 2940 = 86 4360 0 pad 0 )
        96        (because 4360 * 4360 = 19 0096 00 )
```

2.5 Data is often transmitted in 8 bit parcels with the first seven bits being the actual data and the eighth bit being the "parity bit." The parity bit is used to detect errors in transmission. Each bit is either a 0 or a 1. Two techniques for detecting errors are called "odd parity" and "even parity." The transmitter uses the eighth bit to make the eight bit parcel contain an odd number of "1" bits (odd parity) or an even number of "1" bits (even parity). If the transmitter uses odd parity but an even number of "1" bits are received in the transmitted 8 bit parcel, then an ERROR has occurred. If even parity is used, but an odd number of "1" bits are transmitted in the 8 bit parcel then an ERROR has occurred. Write a program to determine if a string of data (of at most 8 characters) contains an ERROR for the indicated parity which is input as ODD or EVEN. The program must also check to see that the string is 8 characters long and contains only "1"s and "0"s. Display the message "ERROR" or "CORRECT" for each input. Example:

```
 INPUT: Enter bits: 00011101
        Enter parity: EVEN
OUTPUT: CORRECT

 INPUT: Enter bits: 11111110
        Enter parity: ODD
OUTPUT: CORRECT

 INPUT: Enter bits: 10101011
        Enter parity: EVEN
OUTPUT: ERROR

 INPUT: Enter bits: 111X1111
        Enter parity: ODD
OUTPUT: ERROR

 INPUT: Enter bits: 110
        Enter parity: EVEN
OUTPUT: ERROR
```

2.6 Write a program that calculates the area of a polygon, using the following formula:

Area = 1/2 * (The absolute value of the following sum):

$$\left|\begin{matrix} X1 & Y1 \\ X2 & Y2 \end{matrix}\right| + \left|\begin{matrix} X2 & Y2 \\ X3 & Y3 \end{matrix}\right| + \ldots + \left|\begin{matrix} Xn\text{-}1 & Yn\text{-}1 \\ Xn & Yn \end{matrix}\right| + \left|\begin{matrix} Xn & Yn \\ X1 & Y1 \end{matrix}\right|$$

where X and Y are coordinates of the vertices in the X-Y plane, and n is the number of vertices. In words, the AREA is equal to one-half the absolute value of the sum of the corresponding determinants of the coordinates of successive vertices. A determinant of the form: $\left|\begin{matrix} X & Y \\ A & B \end{matrix}\right|$ is defined as X*B - A*Y, where X,Y and A,B are two successive vertices. The number of vertices, n, will be entered followed by n successive integer coordinates in the X-Y plane. The area is to be displayed in the form ##.#. Example:

```
INPUT: Enter n: 4
       Enter vertex: 1,0
       Enter vertex: 2,0
       Enter vertex: 2,2
       Enter vertex: 0,4

OUTPUT: AREA =  4.0
```

2.7 Write a program to accept as input a valid MONTH, DAY, and YEAR between the years 1700 and 1998 inclusive and will display the date of the day before the given date and the date after the given date. A leap year occurs in every year that is divisible by 4 except in those years also divisible by 100. MONTH, DAY, and YEAR will be input as positive integers. The output must be displayed in the form MM-DD-YYYY, with MM and DD displayed without leading zeroes. Example:

```
INPUT: Enter month, day, year: 3, 1, 1800

OUTPUT: 2-28-1800
        3-2-1800
```

2.8 At the University of South Florida, a student may be dismissed at the end of a semester for a low grade point average (GPA). If the student's Cumulative GPA (CGPA) is ever between 0 and 0.999 then he is immediately dismissed. If the student's cumulative GPA is between 1.0 and 1.999 for two consecutive semesters then the student is dismissed. The GPA is determined by assigning points to grades: A-4, B-3, C-2, D-1, F-0. A grade is assigned to each class a student takes. Each class gives from 1 to 5 credit hours. For a given semester the total number of points earned is obtained from summing the points earned for each class, which is obtained by multiplying the grade number by the number of credit hours for that class. The GPA is obtained by dividing the total points earned by the total number of credit hours taken in that semester. The CUMULATIVE GPA is similarly obtained from dividing the total points earned for all semesters by the total number of credit hours taken in all semesters.

Write a program to accept at most 8 semesters of 4 grades with the number of credit hours for each class grade. The program then calculates and displays the student's semester GPA and cumulative GPA in the form #.###, rounded to the nearest 0.001. If the student is dismissed due to a low GPA, then display the message "STUDENT IS DISMISSED" and then end the program. The example INPUT/OUTPUT below is illustrated by calculations within parenthesis that are not a part of the INPUT/OUTPUT.

Example:

```
 INPUT: Enter grade, credits: A, 3        (points = 4x3 = 12)
        Enter grade, credits: B, 3        (points = 3x3 =  9)
        Enter grade, credits: C, 2        (points = 2x2 =  4)
        Enter grade, credits: F, 4        (points = 0x4 =  0)
                                          (total =       25)

OUTPUT:  GPA= 2.083          (because 25 / (3+3+2+4) = 2.0833)
        CGPA= 2.083

 INPUT: Enter grade, credits: F, 5
        Enter grade, credits: D, 2         (points = 1x2 = 2)
        Enter grade, credits: F, 4
        Enter grade, credits: F, 5

OUTPUT:  GPA= 0.125
        CGPA= 0.964           (because (25+2) / (12+5+2+4+5))
        STUDENT IS DISMISSED
```

2.9 Given the elements and their corresponding potentials in DATA statements as shown below, write a program to display the names of all pairs of elements which could be used to produce a battery with voltage equal to the desired VOLTAGE plus or minus the TOLERANCE. The desired battery VOLTAGE and a TOLERANCE will be entered as non-negative real numbers. A battery can be produced if the "difference" in half-reaction oxidation potentials of two elements is less than or equal to the TOLERANCE. If there are no pairs that can form a battery, print the message: "NO BATTERY CAN BE FORMED". The acceptable pairs must be printed with the first element starting in column1, the second element in column 12, and the difference in Potential in column 23 in the form #.##. The order of elements is not important. If there are more than eight acceptable pairs then the first eight should be printed and the user is then prompted with the message, "PRESS ANY KEY FOR MORE". The user then presses a key to see the next eight (or less). If there are still more, then the same process should be repeated until all the pairs have been displayed.

Table of half-reaction oxidation potentials:

```
DATA  ELEMENT  POTENTIAL
---- --------- ---------
DATA "LITHIUM",+3.05
DATA "SODIUM",+2.71
DATA "ZINC",+0.76
DATA "IRON",+0.44
DATA "TIN",+0.14
DATA "IODINE",-0.54
DATA "SILVER",-0.80
DATA "MERCURY",-0.85
DATA "BROMINE",-1.09
DATA "CHLORINE",-1.36
```

Example:

```
 INPUT: Enter Desired Voltage, Tolerance: 5, 1.5

OUTPUT: (in any order)
        LITHIUM    IODINE     3.59
        LITHIUM    SILVER     3.85
        LITHIUM    MERCURY    3.90
        LITHIUM    BROMINE    4.14
        LITHIUM    CHLORINE   4.41
        SODIUM     SILVER     3.51
        SODIUM     MERCURY    3.56
        SODIUM     BROMINE    3.80

        PRESS ANY KEY FOR MORE
 INPUT: (Press any key)
OUTPUT: (continued in any order)
        SODIUM     CHLORINE   4.07
```

2.10 Three local cross country teams compete in a double dual race. Each team consists of seven runners, but only the first five finishers of a team contribute to that team's score. As the runners cross the finish line, gasping for breath, the judge writes the INITIAL of the runner's team name and the NUMBER indicating the runner's finishing position, e.g. 1 for 1st, 2 for 2nd and so on. To find the score for teams A and B and to decide which of the two wins, the scorekeeper temporarily eliminates all of C's positions, and then repositions the runners from A and B into places 1 through 14. The team's score consists of the sum of the places of their first five runners. The lower team score wins. If there is a tie then the team whose sixth runner crossed the finish line first is the winner.

Write a program that computes the score for each pair of three teams and determines the winner of each pair (pairs may be displayed in any order). The program must allow the user to assign all 21 runners' team INITIAL to finishing places. Team initials can be any letter in the alphabet. Example:

```
INPUT: Place 1: A
       Place 2: B
       Place 3: A
       Place 4: B
       Place 5: A
       Place 6: B
       Place 7: C
       Place 8: C
       Place 9: C
       Place 10: C
       Place 11: B
       Place 12: A
       Place 13: C
       Place 14: B
       Place 15: C
       Place 16: B
       Place 17: A
       Place 18: A
       Place 19: C
       Place 20: B
       Place 21: A

OUTPUT: TEAM A: 28 POINTS
        TEAM B: 28 POINTS
        TEAM B WINS!

        TEAM A: 25 POINTS
        TEAM C: 31 POINTS
        TEAM A WINS!

        TEAM B: 24 POINTS
        TEAM C: 31 POINTS
        TEAM B WINS!
```

```
Example of
repositioning
teams A and B:

 1: A
 2: B
 3: A
 4: B
 5: A
 6: B
 7: B
 8: A
 9: B
10: B
11: A
12: A
13: B
14: A
```

3.1 Write a program to put a set of N real numbers in numerical order, smallest to largest. However, the magnitude of the digits, from smallest to largest, is defined to be 0,8,1,2,5,4,3,9,7,6. Therefore, 810 is less than 172, and -1.5 is less than -8.5. Example:

```
 INPUT: Enter N: 5
        Enter #: 61.2
        Enter #: 801
        Enter #: -5.98
        Enter #: -4.5
        Enter #: 99.99

OUTPUT: -4.5
        -5.98
        99.99
        61.2
        801
```

3.2 A bank can make change for a given amount of money in many different ways. For example, there are 4 ways to make change for 10 cents: 10 pennies, 5 pennies and 1 nickel, 2 nickels, or 1 dime. Write a program to display the total number of ways that change can be made with an input AMOUNT of money using pennies, nickels, dimes, and quarters. AMOUNT will be input in dollars and will be no greater than 2.00. Example:

```
 INPUT: Enter AMOUNT: 0.16

OUTPUT: 6
```

3.3 Write a program to determine if a given point and/or a given cube in space lie/lies "inside" another given cube (CUBE2). The term "inside" includes points or faces shared. The two coordinates of a diagonal of each of the cubes will be input as real numbers in the X-Y-Z coordinate system. Each cube will have its edges parallel to either the X, Y, or Z plane. Output must contain two sentences of the form:

POINT / 1ST CUBE LIES / DOES NOT LIE INSIDE 2ND CUBE

Example:

```
 INPUT: Enter point: 3,4,5
        Enter cube1 diagonal point1:  0,0,0
        Enter cube1 diagonal point2:  3,3,3
        Enter cube2 diagonal point1: -1,0,0
        Enter cube2 diagonal point2:  4,4,4

OUTPUT: POINT DOES NOT LIE INSIDE 2ND CUBE
        1ST CUBE LIES INSIDE 2ND CUBE
```

3.4 Given a set of N letters (where N is between 2 and 5, inclusive), display all the DISTINGUISHABLE permutations of the letters, in alphabetical order. Also display the total number of distinguishable permutations. Note that some of the letters entered could be the same. Example:

```
 INPUT: Enter letters: CABA

OUTPUT: AABC
        AACB
        ABAC
        ABCA
        ACAB
        ACBA
        BAAC
        BACA
        BCAA
        CAAB
        CABA
        CBAA
        TOTAL= 12
```

3.5 Write a program to print a snake (a trail of 25 asterisks '*') centered on the screen. Upon hitting appropriate keys (I--up, J--left, K--right, and M--down), the snake's head moves in the appropriate direction while the rest of the snake slithers along the same right angle paths. The snake is to move CONTINUOUSLY in the designated direction UNTIL a new directional key is hit. The snake will be 25 asterisks long throughout the entire run; Do not leave a sketched path. The snake cannot go backwards, e.g. if it is going to the right, then its next direction cannot be to the left. The snake continues moving until it runs into itself or it runs off the screen or a non-directional key is pressed.

3.6 Write a program to solve a pair of linear equations entered as strings without spaces. The equation will be in one of two forms: AX+BY=C or AX+BY+C=0 where A, B, and C are integers. Assume that the constant value "1" for A and B will be omitted. (e.g. "X" or "Y" instead of "1X" or "1Y"). For "-1", only a "-" will be used. All unnecessary "+" symbols will be omitted. If a unique solution exists then display the solution in the following format:

XSOLUTION= #.# YSOLUTION= #.#

If no solution exists then print NO UNIQUE SOLUTION EXISTS.
Example:

INPUT: Enter equation 1: **X+Y=0**
Enter equation 2: **2X-3Y-4=0**

OUTPUT: **XSOLUTION= 0.8 YSOLUTION= -0.8**

3.7 Write a program to find all semi-perfect numbers less than 35. A number is said to be semi-perfect if there exists a subset of proper divisors of that number that sum to itself. For example, 12 is semi perfect because 1+2+3+6=12 or 2+4+6=12. Next to each semi-perfect number must be the example(s) of how it is semi-perfect. The example(s) are to be ascending order by the factors. If more than one example exists for a semi-perfect number, then display the example with the fewest addends first; if two have the same number of addends, then display the one containing the smallest addend first. Output must be of the following format:

OUTPUT:

SEMI #	**EXAMPLE(S)**
6	**1 + 2 + 3**
12	**2 + 4 + 6**
12	**1 + 2 + 3 + 6**
:	:
:	:

3.8 Write a program to keep score for a bowler. Input will be 10 frames of numbers. Output will be the scoring of each frame, as shown below in the example. The standard method of scoring will be used in this program.

In each frame the bowler has at most two chances to roll down all 10 pins. If the bowler does not knock down all the pins in that frame, then his score is added to the number of pins that he previously knocked down. (Note: the bowler's score starts out as 0).

For the first 9 frames, if the bowler knocks all 10 pins down on the 2nd roll (indicated by a /) then his score for that frame is his previous score + 10 + the number of pins that he knocks down on his next roll in the next frame. If he rolls all the pins down on his 1st ball of a frame (indicated by an X) then he gets 10 + the total number of pins that he knocks down on the next 2 rolls. A new frame is started after 2 balls are rolled or all ten pins are knocked down on the 1st ball.

In the 10th frame, the bowler is allowed at most three rolls, depending upon his first and second rolls. If he gets all the pins down on the 1st roll then his score will be his previous score plus 10 plus the number of pins he knocks down on his next 2 rolls. If he knocks all 10 pins down after 2 rolls, his score will be his previous score + 10 + the number of pins he knocks down on his next roll. Input will be 10 frames, each separated by comma or a space. For the output, each frame input is right justified and the score is left justified in each frame. Example:

INPUT: Enter frames: **7/,X,62,X,X,8/,63,X,X,X9-**

Note: 62 indicates that 6 pins were knocked down on 1st roll
and 2 pins were knocked down on 2nd roll;
/ indicates all remaining pins knocked down on 2nd roll;
X indicates 10 pins knocked down on the 1st roll;
- indicates 0 pins knocked down.

OUTPUT:
```
-1- -2- -3- -4- -5- -6- -7- -8- -9- -10-
---!---!---!---!---!---!---!---!---!---!
 7/!  X! 62!  X!  X! 8/! 63!  X!  X!X9-!
20 !38 !46 !74 !94 !110!119!149!178!197!
-----------------------------------------
```

3.9 Write a program to convert a real number fraction in base M to a real number fraction in base N, where M and N are input as integers between 2 and 16 inclusive. The base M number input will have one whole number on the left of the radix point, ".", and at most seven whole numbers on the right of the radix point. The judging criterion guarantees that the output base N number will always have one whole number on the left of the radix point. Furthermore, there will be at most seven whole numbers, NDigits, on the right of the radix point as determined by the following formula:

```
(1/N) ** NDigits  less than or equal to  (1/M) ** MDigits

  where N, M are input as bases;
  ** represents "raised to the power of"
  MDigits is the number of digits to the right of the input
  fraction;
  NDigits is the smallest integer that satisfies the equation.
```

For example:

```
  INPUT: Enter M, N, #: 2, 5, 1.11011

  (1/5)**NDigits  less than or equal to  (1/2)**5
  (1/5)**NDigits  less than or equal to  1/32
           NDigits = 3 because
     1/5 ** 2 = 1/25   and   1/5 ** 3 = 1/125
  Therefore 3 digits will be to the right of the output
  number, with the 3rd digit ROUNDED TO THE NEAREST
  NUMBER based upon the 4th digit.

  .11011 base 2 = ( 1/2 + 1/4 + 1/16 + 1/32) base 10
                = ( .84375 ) base 10

  .84375 base 10 = ( .4102) Base 5
                 = ( 4/5 + 1/25 + 0/125 + 0/625) base 10

 OUTPUT: 1.410    (Note: 02 ROUNDED DOWN TO 0 because
                  2 is less than N/2=2.5)


  INPUT: Enter M, N, #: 16, 10, 8.FC2

  (1/10) ** NDigits  less than or equal to (1/16)**3
  Therefore, NDigits is 4.

  .FC2 base 16 = 15/16 + 12/256 + 2/4096 = .98486 base 10

 OUTPUT: 8.9849   (Note: 86 ROUNDED UP TO 9 because
                  6 is greater than or equal to N/2)
```

3.10 Write a program to display the composition of two polynomials, given as input. A polynomial is a mathematical expression consisting of constants multiplied by variables raised to a non-negative integral power. The following are examples of polynomials, where ** stands for "to the power of:"

```
p(x)=5x**2 + 4x - 1

q(x)=x**2 - 6
```

The composition of p of q, p(q(x)), is

```
5*(x**2 - 6)**2 + 4*(x**2 - 6) - 1
5*(x**4 - 12x**2 + 36) + 4*(x**2 - 6) - 1
5x**4 - 60x**2 + 180 + 4x**2 - 24 - 1
5x**4 - 56x**2 + 155
```

Two polynomials will be input. First the ORDER (highest power) of the polynomial is entered as an integer from 0 to 5. The coefficient of each term is entered starting with the highest power, then next highest power, down to the 0th power. The second polynomial is entered the same way.

The output will consist of the composite function of the first of the second polynomial p(q(x)), and then the composite function of the second of the first polynomial q(p(x)). Starting from the highest ORDER of the composite function, the output will be of the following format:

```
AX**N + BX**(N-1) + ... + CX**1 + DX**0.
```

For example:

```
 INPUT: Enter the ORDER of p(x): 2
        Enter coefficient for x**2: 5
        Enter coefficient for x**1: 4
        Enter coefficient for x**0: -1

        Enter the ORDER of q(x): 2
        Enter coefficient for x**2: 1
        Enter coefficient for x**1: 0
        Enter coefficient for x**0: -6

OUTPUT: P(Q(X))= 5X**4 + 0X**3 + -56X**2 + 0X**1 + 155X**0
        Q(P(X))= 25X**4 + 40X**3 + 6X**2 + -8X**1 + -5X**0

 INPUT: Enter ORDER of p(x): 0
        Enter coefficient for x**0: 9

        Enter the ORDER of q(x): 1
        Enter the coefficient of x**1: -4
        Enter the coefficient of x**0: 3

OUTPUT: P(Q(X))= 9X**0
        Q(P(X))= -33X**0
```