IE5600 – Applied Programming for Industrial Systems

AY 2023/24 Semester 1

Practical Lab 03 – Iterative Control Flow

Section B – Programming Exercises

Question B1 – Multiplication Table (Basic)

Write a program that asks user to input a positive integer. Thereafter, the program should print out the multiplication table of the integer from 1 to 10.

Use the `while` statement and `for` statement to write the program and then briefly explain which statement is more suitable for solving this problem?

Sample Input	Sample Output
2	2, 4, 6, 8, 10, 12, 14, 16, 18, 20
13	13, 26, 39, 52, 65, 78, 91, 104, 117, 130

Question B2 – Perfect Square (Basic)

Write a program that asks the user to input a positive integer from 1 to 100. Thereafter, the program should print out whether the integer is a perfect square. A perfect square is the square of an integer.

Sample Input	Sample Output
14	Not a perfect square
25	Perfect square
40	Not a perfect square
81	Perfect square

Question B3 – Find the Last Digit (Basic)

Write a program to print out the last digit, i.e., the rightmost digit, of any positive integer input by the user.

You are **NOT** allowed to treat the input as string.

Sample Input	Sample Output
1	1
5	5
17	7

258	8

Question B4 – Find the First Digit (Basic)

Write a program to print out the first digit, i.e., the leftmost digit, of any positive integer input by the user.

You are **NOT** allowed to treat the input as string.

Sample Input	Sample Output
1	1
5	5
17	1
258	2
45644478	4
56789543257899	5

Question B5 – Find Any Digit (Basic)

Write a program to print out the n^{th} digit, starting from the left, of any positive integer input by the user. If the requested n^{th} digit input by the user is invalid, the program should print out an appropriate error message.

You are **NOT** allowed to treat the input as string.

Sample Input	Sample Output
2345, 0	Error: Digit position must be greater than 0
2345, 1	2
2345, 2	3
2345, 3	4
2345, 4	5
2345, 5	Error: Digit position is more than the number of digits.
56789543257899, 4	8

Question B6 – The World of Triangles (Basic)

Write a program to draw a triangle with asterisks on the screen. The program should:

1. Prompt the user to input the base in units.

2. Compute the height of the triangle using the formula $(base + 1)/2$ and output the height to user.

3. Draw a triangle of the respective base and height on the screen using asterisks.

4. For example, if user input a base of 9 units, the program should compute the height as 5 units and then draw this triangle using asterisks. See the sample output below.

Sample Program Run:

```
Enter Base of Triangle (Odd Number Integer, Minimum is 3 units) = 9
You have requested a triangle with a base of 9 units :)
This triangle has a computed height of 5 units :)
Here is your triangle...

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

Question B7 – Prime Factorisation (Basic)

Prime factors of a positive integer greater than 1 are the prime numbers that divide that integer exactly, without leaving a remainder. The process of finding these prime numbers is called prime factorisation. A prime number is itself defined as a positive integer greater than 1 that has no positive divisors other than 1 and itself.

Write a program to generate the prime factors of any positive integer greater than 1.

Sample Input	Sample Output
2	2
48	2 x 2 x 2 x 2 x 3
255	3 x 5 x 17
9001	9001
67786531	17 x 443 x 9001

Question B8 – Guess the Computer's Age (Intermediate)

Write a program that generates a random positive integer between 1 and 100 that represents the computer's age.

The program should then ask user to guess the computer's age. If the user guesses the correct age, print out a congratulatory message. Otherwise, print out a message to tell the user whether the guess is too small or too big. The program should also keep track of the number of attempts that the user has taken to guess the correct age. This information should be printed out as part of the congratulatory message after a correct guess has been made.

You may assume that the random positive integer may be generated by calling some system specific code. In Python, `random.randrange(n)` will return a random integer in the range [0, n), or equivalently [0, n-1].

The program should continue to generate a new random positive integer after the user has correctly guessed the current age until the user chooses to exit.

Sample Program Run:

```
Enter your guess of the computers age = 50
Your guess is too big!

Enter your guess of the computers age = 25
Your guess is too small!

Enter your guess of the computers age = 40
Your guess is too big!

Enter your guess of the computers age = 30
Your guess is too small!

Enter your guess of the computers age = 35
Your guess is too big!

Enter your guess of the computers age = 33
Your guess is too small!

Enter your guess of the computers age = 34
Congratulations! You have guessed the computers age correctly
You took 7 attempt(s)

Do you want to continue with the game (Y: Yes, N: No) =
```

Question B9 – Pascal's Triangle (Intermediate)

Pascal's triangle is a triangular array of binomial coefficients arranged in a triangle.

To construct a Pascal's triangle, we begin by numbering the first row as $n = 0$. The second row would be $n = 1$, the third row would be $n = 2$, etc. Each row contains $n + 1$ elements with the first element numbered as $k = 0$. The second element would be $k = 1$, the third element $k = 2$ until the last element, which is $k = n + 1$.

On row 0, we simply write the number one. To derive the elements for the subsequent rows, we use this algorithm – add the number directly above and to the left with the number directly above and to the right of a specific element to find the new number for that element. If either the number to the left or right is not present, we will substitute a zero in its place.

A Pascal's triangle of 6 rows is shown in the figure below. The first element, i.e., $k = 0$, in the sixth row, i.e., $n = 5$, is derived as $0 + 1 = 1$. The second element, i.e., $k = 1$, in the sixth row is derived as $1 + 4 = 5$.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Mathematically, the number of each element is calculated with the same formula for calculating combinations:

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

For example, 5 *choose* 3, i.e., the sixth row ($n = 5$) fourth element ($k = 3$) in the figure, would be:

$$\binom{5}{3} = \frac{5!}{3!\,(5-3)!} = \frac{5!}{3! \times 2!} = \frac{5 \times 4 \times 3 \times 2 \times 1}{(3 \times 2 \times 1) \times (2 \times 1)} = \frac{5 \times 4}{2 \times 1} = 10$$

To simplify the computation of 5 *choose* 3, we can actually cancel out the $(3 \times 2 \times 1)$ in the denominator from the numerator. This will give us an easier formula to program in C:

$$\binom{n}{k} = \frac{\prod_{i=k+1}^{n} i}{(n-k)!}$$

Using the simplified formula, 5 *choose* 3 would be computed as:

$$\binom{5}{3} = \frac{\prod_{i=3+1}^{5} i}{(5-3)!} = \frac{\prod_{i=4}^{5} i}{2!} = \frac{4 \times 5}{2 \times 1} = 10$$

Write a program that prompts the user for the number of rows required. Then output the Pascal's triangle with the required number of rows. The maximum number of rows that you need to handle is nine rows. If you wish to go beyond nine, don't worry if the shape of the triangle is not symmetrical.

Hints: The product operator for the product of a sequence, i.e., Π, can be computed using a similar algorithm as that for computing factorial.

Sample Program Run:

```
Enter the Number of Rows Required = 9
You have requested a Pascal Triangle of 9 rows :)

                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5  10  10   5   1
    1   6  15  20  15   6   1
  1   7  21  35  35  21   7   1
1   8  28  56  70  56  28   8   1

In [90]: runfile('D:/Dropbox/Teaching - NUS FOE - CMS/IE5600 - AY202122S2/Practical Lab/Week 05/src/pe03-b9.py', wdir

Enter the Number of Rows Required = 20
You have requested a Pascal Triangle of 20 rows :)

                                    1
                                  1   1
                                1   2   1
                              1   3   3   1
                            1   4   6   4   1
                          1   5  10  10   5   1
                        1   6  15  20  15   6   1
                      1   7  21  35  35  21   7   1
                    1   8  28  56  70  56  28   8   1
                  1   9  36  84 126 126  84  36   9   1
                1  10  45 120 210 252 210 120  45  10   1
              1  11  55 165 330 462 462 330 165  55  11   1
            1  12  66 220 495 792 924 792 495 220  66  12   1
          1  13  78 286 715 1287 1716 1716 1287 715 286  78  13   1
        1  14  91 364 1001 2002 3003 3432 3003 2002 1001 364  91  14   1
      1  15 105 455 1365 3003 5005 6435 6435 5005 3003 1365 455 105  15   1
    1  16 120 560 1820 4368 8008 11440 12870 11440 8008 4368 1820 560 120  16   1
  1  17 136 680 2380 6188 12376 19448 24310 24310 19448 12376 6188 2380 680 136  17   1
1  18 153 816 3060 8568 18564 31824 43758 48620 43758 31824 18564 8568 3060 816 153  18   1
1  19 171 969 3876 11628 27132 50388 75582 92378 92378 75582 50388 27132 11628 3876 969 171  19   1
```

Question B10 – Greatest Common Divisor (Intermediate)

The greatest common divisor (gcd) of two positive integers is the largest positive integer that divides the two numbers without a remainder. Write a program that prompts user to input two positive integers and then compute the gcd.

You are **NOT** allowed to use any function from the `math` module.

Post Exercise Thoughts: How long does your program take to compute the gcd for two numbers? If it takes longer than the blink of your eyes, try to find a faster algorithm ☺

Sample Input	Sample Output
48,18	6
1235, 255	5
55546364, 5656	404

Question B11 – Least Common Multiple (Intermediate)

The least common multiple (lcm) of two positive integers is the smallest positive integer that is divisible by both numbers. Write a program that prompts user to input two positive integers and then compute the lcm.

You are **NOT** allowed to use any function from the `math` module.

Sample Input	Sample Output
4, 2	4
234, 123	9594
8968, 5687	51001016

Question B12 – Collatz Conjecture (Advanced)

The Collatz Conjecture asks whether repeating two simple arithmetic operations will eventually transform every positive integer into one. It concerns sequences of integers in which each term is obtained from the previous term as follows:

- If the previous term is even, the next term is one half of the previous term.
- If the previous term is odd, the next term is 3 times the previous term plus 1.

The conjecture is that these sequences always reach 1, no matter which positive integer is chosen to start the sequence. Thus, this conjecture is also commonly known as the $3n + 1$ problem or sequence.

a) Write a program that prompts user to input a positive integer and then print out its $3n + 1$ sequence using either a `while` or `for` statement.

Sample Input	Sample Output
3	3, 10, 5, 16, 8, 4, 2, 1

19	19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
21	21, 64, 32, 16, 8, 4, 2, 1
16	16, 8, 4, 2, 1

b) Write another program that performs the same tasks as (a) but using recursion.

c) Suppose you would like to prove that the Collatz Conjecture holds for all positive integer numbers up to 10^{12}. Research on the Internet for a viable algorithm and then attempt to implement the algorithm in Python.

Try to run your program and observe if you could find a positive integer number that defies the Collatz Conjecture.

As of 2020, the conjecture has been checked by computer for all values up to $2^{68} \approx 2.95 \times 10^{20}$. Obviously, this computer evidence is not sufficient to prove that the conjecture is true for all starting values. If you could find a way to solve this problem, there is a pot of gold awaiting you at the end of the rainbow :)