CS101- Algorithms and Programming I

Lab 04

Lab Objectives: While Loops

- ❏ For all labs in CS 101, your solutions must conform to these CS101 style guidelines (rules!)
- ❏ Create a Lab04 workspace (i.e. the folder H:\private\cs101\lab04). This assignment has parts a, b, c & d, each of which should be placed in a separate project within the same Lab04 workspace. Note: only one project is active at a time. To work (Build/Run) a different project, right click on the project's name and select "Set as active project".
- ❏ You can only use "while" loops for this lab assignment. You cannot use "for", "do-while" or any other repetition method.

---

a. Create a new project Lab04a. Write a program that calculates the factorial of a given number. First, it asks for an integer number x between [0 – 13] and calculates its factorial. The program should run until user's input is an invalid number (outside of interval). Are your results correct? Explain.

A sample run is shown below:

```
Please enter a number [0-13]: 0
0! = 1
Please enter a number [0-13]: 5
5! = 120
Please enter a number [0-13]: 2
2! = 2
Please enter a number [0-13]: 10
10! = 3628800
Please enter a number [0-13]: 12
12! = 479001600
Please enter a number [0-13]: 4
4! = 24
Please enter a number [0-13]: -1
Goodbye!
```

b. Create a new project Lab04b. Your program will input one positive integer and calculate its prime divisors. Note that prime divisors of a number cannot be greater than this value and cannot be smaller than 2.
You may assume the user enters valid inputs, you are not expected to validate the input.

Sample run:

```
Enter an integer: 60
Prime divisors of 60: 2 3 5
```

Another sample run:

```
Enter an integer: 123
Prime divisors of 123: 3 41
```

c. Create a new project Lab04c. Your program prompts the user to enter a character, ***ch***, and an int value, ***width***, and then print out a triangle formed using ***ch*** characters having a height of ***width*** characters. For example, if the user enters "*" and 5, it should print,

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

d. Create a new project Lab04d. In this part, you are going to compute arctan(x) in radians. The following formula approximates the value of arctan(x) using Taylor series expansion:

$$\tan^{-1}(x) = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1}, \qquad x \in (-1, +1)$$

Depending on the number of terms included in the summation the approximation becomes more accurate.

Your program will input from the user the value of x, which should be in the range (-1,1) and also the precision of the summation. You are going to calculate how many terms of the series are needed to approximate the sum up to the given precision for the given x value.

Since it is an alternating series, the absolute error of the approximation of arctan(x) by the partial sum, $\sum_{k=0}^{n}(-1)^k \frac{x^{2k+1}}{2k+1}$,

$$\left|\tan^{-1}(x) - \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{2k+1}\right| = \left|\sum_{k=n+1}^{\infty} (-1)^k \frac{x^{2k+1}}{2k+1}\right| \leq \frac{|x|^{2n+3}}{2n+3}$$

is bounded by $\frac{|x|^{2n+3}}{2n+3}$, that is the absolute value of the (n+1)st term in the series. If this absolute error of the approximation is desired to be less than some precision > 0, then it is sufficient to set n to be smallest integer n such that $\frac{|x|^{2n+3}}{2n+3} \leq precision.$

In your program, calculate the sum repeatedly until the absolute value of the term becomes less than or equal to precision. At each step, print the approximated value of the sum so that you can see the progression. Examine the sample output carefully.

You may assume the user enters valid inputs, you are not expected to validate the input.

Hints:

- ❏ You can use Math.pow() method to raise x to (2k+1)th value, e.g. Math.pow(3, 2) raises 3 to its 2nd power.
- ❏ You can use Math.abs() method for the absolute value of x.

A sample run is shown below:

```
Enter x: 0.76
Enter precision: 0.0001

Current sum: 0.76
Current sum: 0.6136746666666667
Current sum: 0.6643851741866666
Current sum: 0.6434634676555581
Current sum: 0.6528624280829557
Current sum: 0.6484206502751572
Current sum: 0.6505915179274363
Current sum: 0.6495048105256074
Current sum: 0.6500586477567513
Current sum: 0.6497724246756961
Current sum: 0.6499220021319214
Current sum: 0.6498431188835287

Computed with 1.0E-4 precision in 12 steps.
arctan(0.76) = 0.6498431188835287 radians.
```