EN 001203 Computer Programming

L004: Iteration Exercises

Faculty of Engineering, Khon Kaen University

Submission: https://autolab.en.kku.ac.th

==========

* Submit an answer to a question with a file with `txt` extension. E.g., an answer for Q1 should be submitted in a text file "`Q1.txt`"

* Submit a program to a (programming) problem with a file with `cpp` extension. E.g., a program for P2 should be named "`P2.cpp`"

* All answers and programs must be packaged together in a tar file.

* Each question or problem is worth 60 points. The 240 points are counted as full score. Students are encouraged to work on as many problems as they can, but seeing 240 points could make any programming instructor super content already.

==========

"I learned that courage was not the absence of fear, but the triumph over it. The brave man is not he who does not feel afraid, but he who conquers that fear."

Nelson Mandela

P1. Simple while. Write a program to ask a use for a starting financial balance, an annual return factor, and target. The program calculates a new balance that

a new balance = a current balance * return factor.

Then, the program prints out the new balance and repeat the process until a balance reaches the target.

[Hint: a while statement may be more suitable when a number of iterations is determined by a condition.]

Have the program interact exactly like what shown in the examples. Starting balance, annual return factor, and target are inputted by a user.

Output example: when inputting starting balance, annual return, and target as 10000, 1.1, and 20000

```
balance: 10000
```

```
factor: 1.1
target: 20000
Year 1: balance = 11000
Year 2: balance = 12100
Year 3: balance = 13310
Year 4: balance = 14641
Year 5: balance = 16105.1
Year 6: balance = 17715.6
Year 7: balance = 19487.2
Year 8: balance = 21435.9
```

P2. Simple for. Write a program to ask a use for a starting financial balance, an annual return factor, and a number of years. The program calculates a new balance that

a new balance = a current balance * return factor.

Then, the program prints out the new balance and repeat the process for a number of years as user specified.

[Hint: a for statement may be more suitable when a number of iterations is fixed.]

Have the program interact exactly like what shown in the examples. Starting balance, annual return factor, and a number of years are inputted by a user.

Output example: when inputting starting balance, annual return, and a number of years as 100000, 1.02, and 4

```
balance: 100000
factor: 1.02
years: 4
Year 1: balance = 102000
Year 2: balance = 104040
Year 3: balance = 106121
Year 4: balance = 108243
```

P3. Factorial. Revised that a factorial of a positive integer n! = n · (n – 1) · (n – 2) · · · 4 · 3 · 2 · 1. Write a program to take a positive integer from a user and compute its factorial.

Have the program interact exactly like what shown in the examples. A number n is inputted by a user.

Output example: when inputting n as 4

```
n: 4
n! = 24
```

P4. Power series of exponential function. Given a power series of an exponential function $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, write a program to calculate a sequence

$$e^x = \sum_{n=0}^{M} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \cdots + \frac{x^M}{M!}$$

where x and M are taken from a user.

[Hint: function pow of library cmath may be handy.]

Have the program interact exactly like what shown in the examples.

Output example: when inputting x and M as 1 and 10

```
x: 1
M: 10
s = 2.71828
```

P5. Carbon-14 dating. Living organism has a relatively constant ratio of carbon-12 to carbon-14, but unstable carbon-14 will decay (and turn to nitrogen) at a constant rate after an organism dies, while carbon-12 remains constant. The drop in ^{14}C-to-^{12}C ratio can then be used to determine the last time the organism has lived.

Given the half-life of carbon-14 is 5730 years and the ^{14}C-to-^{12}C starting ratio is about 1 to 10^{12} (src: https://www.canadianarchaeology.ca/dating), write a program to take a starting ratio and an instrument sensitivity (the smallest ratio it can detect) and print out the estimated ^{14}C-to-^{12}C ratio at every 5730 year until the ratio goes below the sensitivity.

Have the program interact exactly like what shown in the examples. Both ratio and sensitivity are inputted by a user.

Output example: when inputting ratio (carbon-14 to 10^{12} carbon-12) and sensitivity (carbon-14 to 10^{12} carbon-12) as 1 and 0.001

```
ratio: 1
sensitivity: 0.001
Year 0: ratio = 1
Year 5730: ratio = 0.5
Year 11460: ratio = 0.25
Year 17190: ratio = 0.125
Year 22920: ratio = 0.0625
Year 28650: ratio = 0.03125
Year 34380: ratio = 0.015625
Year 40110: ratio = 0.0078125
Year 45840: ratio = 0.00390625
Year 51570: ratio = 0.00195312
```

Output example: when inputting ratio (carbon-14 to 10^{12} carbon-12) and sensitivity (carbon-14 to 10^{12} carbon-12) as 1.2 and 0.004

```
ratio: 1.2
sensitivity: 0.004
Year 0: ratio = 1.2
Year 5730: ratio = 0.6
Year 11460: ratio = 0.3
Year 17190: ratio = 0.15
Year 22920: ratio = 0.075
Year 28650: ratio = 0.0375
Year 34380: ratio = 0.01875
Year 40110: ratio = 0.009375
Year 45840: ratio = 0.0046875
```

"Honesty is often very hard. The truth is often painful. But the freedom it can bring is worth the trying."

Fred Rogers
