MTH 3300
Fall 2016
E. Fink
January 31, 2017

Homework 2

Directions: Write the following 3 programs, and submit your source code to me via Blackboard, using the template files I have provided. You should send only the source code, in cpp format; do NOT send your .exe files. READ THE INSTRUCTIONS on how to submit your work in the Course Documents section of Blackboard.

This programming assignment is meant to solidify your understanding of basic programming design, use of cin and cout, and use of math library functions. Each of the programs below can easily be completed in about 20-30 lines or less.

---

1) population.cpp

Suppose that you are a demographist, who wants to model the growth of the population of a nation over time. A simple model for this growth is the standard exponential model

$$P = P_0 e^{rt}$$

where P_0 is the initial population at time $t = 0$, r is the relative growth rate in percent per year (expressed as a decimal), t is the time elapsed in years, and P is the population at time t. e, of course, is the base of the natural logarithm ($e \approx 2.718$).

Write a program that prompts the user to enter a value for the initial population. The program should then do the following **three** times: it will ask for a number of years and a relative growth rate; the program will then compute the new population after that many years have elapsed, using the population growth formula provided above. This should be done *cumulatively* – that is, the end population for the first iteration should be used as the initial population for the second iteration, and the end population for the second iteration should be used as the initial population for the third iteration.

So, for example: suppose that the user enters an initial population of 300, a first time period of 4 years, and a first growth rate of 1.2%. Then the population at the end of the first time period would be 314.751, since

$$300 \cdot e^{(0.012)(4)} = 314.751.$$

Suppose then that the second time period was 2.5 years, and the second growth rate is 5%; then the population at the end of the second time period will be

$$314.751 \cdot e^{(0.05)(2.5)} = 356.66.$$

Finally, suppose that the user enters 1 year for the last time period, and 2.1% for the last growth rate; then the population at the end of the last time period will be

$$356.66 \cdot e^{(0.021)(1)} = 364.229.$$

That last number, 364.229, is what the program should display as the final population. (Don't worry about that fact that many of these numbers aren't integers.)

Hints: make sure you try calculating with my sample values by hand before you start programming, to make sure you understand the task! You should Google any mathematical C++ functions that you may need, if we haven't discussed them. Finally, in case you are tempted to figure out a way to get C++ to "repeat itself three times" – don't do that, just write similar code three times over (we'll learn about repetition soon enough).

Specifications: your program must

- clearly ask the user to enter the initial population.
- ask the user three times to enter a number of years, and a growth rate. The program should accept the growth rates input as **percents** (but input without the percent sign – so "3.5%" will be input to the program as just `3.5`).
- compute the **final** population as described above, and clearly display it. You are not required to round this value to an integer.

Challenge (optional): try to write this program using as few declared variables as possible.

---

2) madlibs.cpp

Write a program that plays the following game. First, it will ask the user to enter three different nouns (a *noun* is a word that describes a person, place or thing, like "Baruch" or "clock" or "oxygen"). Then, the program should print out the sentence

"_____ read a _____ about _____s."

with the blanks filled in with the three words, in all 6 possible orders.

For example, if the user entered the words `Evan`, `Bear`, and`Book`, then the program should print out:

```
Evan read a Bear about Books.

Evan read a Book about Bears.

Bear read a Evan about Books.
```

as well as the other three possible orders.

Once you've done that, I encourage you to also do the same thing with another sentence that amuses you more – just make sure that it includes three blanks that can be filled by singular nouns.

Don't try to find a clever way to program the six different orderings – just do this in the simplest way you can think of (so there will probably be a lot of cutting and pasting when you write your code).

Specifications: your program must

- clearly ask the user to enter three nouns. You may assume that each of the words will not contain whitespace (spaces, tabs, newlines, etc.).
- print out the given sentence six times: each time, it should be filled in with the three entered words in a different order. If you like, you can do the same with a second sentence.
- print out the sentences with proper spacing and punctuation – there should be exactly one space between consecutive words, and a period immediately following the last letter of the last word. Don't worry about capitalization.

---

3) triangle.cpp

Write a program that will ask the user to input *three positive numbers in descending order*. The program will then – using a specific procedure! – compute the three angles of the triangle that has those three numbers as sidelengths, in degrees.

You will do this by using the Law of Cosines to find the largest angle, then using the Law of Sines to find the middle angle, and then subtract to find the smallest angle. (There are other ways to proceed – you might say they are easier ways! However, I am **insisting** that you do it in this manner.)

I strongly suggest you look up the Law of Sines and the Law of Cosines in case you've forgotten them. Also, don't forget about converting from radians to degrees. In addition, you should Google the trigonometric and inverse trigonometric functions that you will need for your calculations.

Hints: the largest angle is opposite the largest side, the middle angle is opposite the middle side, and the smallest angle is opposite the smallest side. Also, be *very* careful about order of operations!

Specifications: your program must

- ask the user to enter three side lengths in descending order, each of which can be any positive number – even one with decimals. You may assume that the user obeys – if the user enters sides out of order, or negative numbers, then your program does not need to work. You may also assume that the user enters numbers that are the sidelengths of a real triangle! That is, you don't have to worry about the user entering `100`, `2` and `1`, because no real triangle has those three sidelengths (remember the triangle inequality?).
- correctly calculate the three angles in the triangle, in **degrees**, under the assumptions mentioned above.
- use the strategy I outlined above: Law of Cosines to find the largest angle, Law of Sines to find the middle angle, and subtracting to find the smallest angle.
- print out the three angles in degrees, **each on a separate line**. You don't have to worry about how many decimal places they are output with.

Sample data: if I enter `5`, `4`, and `3` as my sidelengths, I should get `90`, `53.1301` and `36.8699` out as the angles (in some order, and don't worry if the decimals are slightly off). If I enter `4.1`, `2.6`, and `2.4`, I should get `110.106`, `36.5484`, and `33.346` in some order.