Lab 2: Released Wednesday, week 2 and due Tuesday, week 3

Note: This lab gives you all of the instructions for writing your code in Step 2, and then the instructions for compiling in Step 3. You are not required to write all three programs before compiling, feel free to write one, compile and test it, and then move on to the next.

Basic C++ Programming

Step 1: Getting Ready

Recall what environment you set up last week in lab 1. Navigate there, and then (similar to last week), clone the Github repository. This time you'll have your own, so first go to the course Github (https://github.com/ucsb-cs16-1-u22) and make sure you can see a repository named Lab02-yourgithubusername. If you can see that repository, open it up in your browser and go to the green "Code" button near the upper right. Copy the SSH link for this repository (for more instructions on this step, and future steps involving Github, check out https://sites.cs.ucsb.edu/~maradowning/cs16/refs/gitbasics.html).

From here, run the following command in the folder you'd like your Lab02 folder to end up in:
`git clone <link you just copied>`

Check afterwards with `ls` to make sure the repository has been cloned properly, you should see a new folder named Lab02-yourgithubusername. You can run the following command to enter this directory and see the starter code files:
`cd Lab02-yourgithubusername`

Step 2: Edit your C++ Files

This assignment consists of 3 problems, each of which is described below. The first two are worth 30 points each, and the last is worth 40 points. Each should be solved in its own file and all 3 must be submitted for full assignment credit.

Important: Do not plagiarize.

Important: For this lab, you are not allowed to use C++ syntax that has not yet been covered in class. For example, we have not discussed functions or arrays yet, so you are not allowed to use those (or other topics not yet covered) in this lab!

Program 1: descend.cpp
Write a program that takes 3 integer inputs from the user and prints them back in descending order.

Hint: Nested if-else statements can be very helpful here!

Remember: You cannot use techniques we have not yet covered in class to solve this!

A session should look exactly like the following example (including whitespace and formatting), with all manner of different values for the input and output. Note, the first line is the user input and the second line what the program outputs.

```
-2 8 4
8 4 -2
```

Another example:

```
22 -41 22
22 22 -41
```

Program 2: block.cpp
Write a program that takes from the user some number of rows and number of columns, and then prints out a block of characters that is based on these 2 parameters. The program should keep asking the user for input, and printing out the result until the user enters zero (0) for each of the input parameters.

A session should look exactly like the following example (including whitespace and formatting), with all manner of different values for the input and output. Each line printed by the program should include a newline at the end, but have no other trailing whitespace (i.e. no extra space characters at the end of the line).

```
Enter number of rows and columns:
10 10
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
X.X.X.X.X.X.X.X.X.X.
Enter number of rows and columns:
3 7
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
X.X.X.X.X.X.X.
Enter number of rows and columns:
0 0
```

Program 3: pi.cpp

Write a C++ program that approximates the value of the constant pi (π), based on the Leibniz formula for estimating π. The formula is shown below and can go on for some arbitrary n-number of terms:

$$1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots = \pi/4$$

The formula works best for high values of n.

The program takes an input from the user for the value of n, which determines the number of terms in the approximation. The program outputs that calculation according to the formula. You must also include a loop that allows the user to repeat this calculation for new values of π until the user says they want to end the program by issuing an input of 0.

The number of terms is assumed to not include the added 1 in the formula. In other words,

If n = 1, then π = 4 * (1 - (1/3))
If n = 2, then π = 4 * (1 - (1/3) + (1/5))
If n = 2, then π = 4 * (1 - (1/3) + (1/5) - (1/7))
And so on. . .

Here is a “skeleton” program to help you get started:

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
	int terms = 1;
	double pivalue = 0;

	// You also need to do a loop here that keeps asking for a number
	// of terms and then approximates pi!

	// HINT: Part of what we want to do is decide the sign of a number.
	// There are multiple ways to do this. One way we can do this is to
	// use cmath for its pow() function, which calculates x raised to
	// the power y when used like: pow(x, y)

	return 0;
}
```

The program should print a string of text to the terminal before getting each piece of input from the user. A session should look like the following example (including whitespace and formatting), with all manner of different values for the input and output. Note that each line printed by the program should include a newline at the end, but have no other trailing whitespace (i.e. no extra space characters at the end of the line). When the user enters 0, there is no approximation given—the program just ends there. It is OK if your approximation results print with a different number of decimal places as the ones in this example (though you should have at least a couple decimal places no matter what).

```
Enter the number of terms to approximate (or zero to quit):
5
The approximation for Leibniz's Formula is 2.97605 using 5 terms.
Enter the number of terms to approximate (or zero to quit):
1000
The approximation for Leibniz's Formula is 3.14259 using 1000 terms.
Enter the number of terms to approximate (or zero to quit):
0
```

Step 3: Compile your Code

To compile your programs, we will use the g++ command as we described in previous labs. The following 3 commands will compile the 3 source files:

```
g++ -std=c++11 -Wall descend.cpp -o descend
g++ -std=c++11 -Wall block.cpp -o block
g++ -std=c++11 -Wall pi.cpp -o pi
```

If compilation is successful, you will not see any output from the compiler. You can then use the following commands to run your programs:

```
./descend
./block
./pi
```

If you encounter an error, use the compiler hints and examine the line in question. If the compiler message is not sufficient to identify the error, you can search online to see when the error occurs in general.

Remember to re-compile after you make any changes to your C++ source code.

Step 4: Submit your Code

There is a more detailed (and more general) set of instructions on the course webpage, under Quick References: https://sites.cs.ucsb.edu/~maradowning/cs16/refs/gitbasics.html. If you are new to git/github, I suggest reading through this guide as well when you submit your assignment.

Once you have finished your code (or any time you want to save your code to Github and take a break), first run `git status` to see which files you have modified/created. From there, add any files you want to end up on Github by running the following command:

```
git add <filename>
```

For this lab, most likely you will run three add commands:

```
git add descend.cpp
git add block.cpp
git add pi.cpp
```

Please do not add any executable files you created by compiling your code. However, if you do end up accidentally adding them, don't worry about it for now. In later labs I will take off a couple points for adding these files, but in these labs I'll just mention it to make sure you're aware.

Once you've added all the files you want, you will need to commit them. Run:

```
git commit -m "Your commit message here"
```

Please write a descriptive commit message within the quotes, something like "Finished Lab02" if you are done, or something more specific if you're pushing code partway through.

Finally, run:

```
git push
```

to push your changes to Github. Once you've run this command, make sure to go to your repository in your web browser and check that the updated files are there (you might have to reload).

Finally, once you're completely finished with the lab, fill out the Lab02 submission on Gradescope. It only asks you for a list of people you discussed the lab with (remember, discussion of the lab is ok but do not give classmates answers), the number of hours you spent, and has a checkbox to indicate that you're done. I will use this as an indication that your code is pushed to Github and I can begin grading.