Lab 02: Basic C++ Programming

Assigned: *Tuesday, October 13th, 2020*
Due: *Monday, October 19th, 2020*
Points: *100 (normalized to 100 in gradebook)*

- There is NO MAKEUP for missed assignments.
- We are strict about enforcing the LATE POLICY for all assignments (see syllabus).

Introduction

The assignment for this week will utilize concepts of input and output and simple control flow (Chapter 2 in the book).

Step 1: Getting Ready

If you have not gotten a CoE account yet, please do so as soon as possible! Open a terminal window and log into the correct machine - refer to the instructions in Lab 00, if you forgot how to do that.

Start by changing into your CS 16 directory:

```
$ cd cs16
```

And then create and change into the lab02 directory:

```
$ mkdir lab02

$ cd lab02
```

Remember that at any time, you can check what directory you are currently in with the command **pwd**.

Step 2: Create and Edit Your C++ Files

For a reminder on how to open and use a text editor to create and edit new source files, refer back to Lab 01.

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 three must be submitted for full assignment credit. These exercises are inspired by ones from the textbook (in Ch. 2) - but they are NOT the same, so follow the instructions on THIS sheet carefully.

IMPORTANT: **Do NOT plagiarize**. We will be able to tell!

IMPORTANT: **Follow the directions carefully below or you may lose points**: you can probably make these programs run with different approaches: but I want you to create them according to the directions.

IMPORTANT: **You are NOT allowed to use C++ code that has not been covered in class**. For example, we have not discussed functions or arrays as of yet in class, and you are not allowed to use those (and other not-covered topics) in this lab!

Program 1: DESCEND.CPP

Write a program that takes 3 integer inputs from a user and prints them back in descending order.
Hint: Nested if-else statements will be very helpful here!
Remember: You cannot use techniques we have not covered in class to solve this!!

A session should look ***exactly*** like the following example (including whitespace and formatting - although note that there is no whitespace at the end of each of these lines), with all manners of different numbers for inputs and the output – note the first line is the user input and the second line is what the program outputs:

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

OR:

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

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).

Program 2: BLOCK.CPP

Write a program that takes input from a user for number of rows and number of columns and 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 for each of the input parameters*.

A session should look ***exactly*** like the following example (including whitespace and formatting - although note that there is no whitespace at the end of each of these lines), with all manners of different numbers for inputs and the output:

```
$ ./block
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:
9 9
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:
4 7
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:
9 3
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:
0 0
```

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).

Program 3: PI.CPP

Write a C++ program that approximates the value of the constant π, based on both the Leibniz formula for estimating π. The formula is shown below and can go on with many (N) elements:

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4}$$

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 of the value of pi. The program then outputs that calculation for the formula. You must also include a loop that allows the user to repeat this calculation for new values N until the user says they want to end the program by issuing an input of 0. Finally, the results must be shown to the 5th decimal place.

The number of terms is assumed to NOT include the added 1 in the formula. In other words,
if N = 1, then pivalue = 4*[1 - (1/3)], and
if N = 2, then pivalue = 4*[1 - (1/3) + (1/5)], and
if N = 3, then pivalue = 4*[1 - (1/3) + (1/5) - (1/7)], etc…

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 need to do something about the formatting requirement here!

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

    // HINT: You can 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 various and different inputs and numbers in the output:

```
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):
50
The approximation for Leibniz's Formula is 3.16120 using 50 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
```

Note that each string printed by the program should include a newline at the end, but no other trailing whitespace (whitespace at the end of the line). Note that when the program exits, when a user enters 0, there is no approximation given - the program just ends there.

Step 3: Compile the Codes

To compile our codes, we will use the same g++ command as we described in previous labs. The following three commands will compile the three source files (in the same order as listed above):

```
$ g++ -o descend descend.cpp
$ g++ -o block block.cpp
$ g++ -o pi pi.cpp
```

If the 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 the relevant files after you make any changes to the C++ code.

Step 4: NEW! Setup Your GitHub Account for This and Future Labs

You are to set up a GitHub account (on https://github.com) and link up to a class account ('organization') for use in future lab assignments for this class.

If you ALREADY have a GitHub account, skip part 1 and go to part 2. Otherwise, read both parts.

Part 1: Steps to getting a new account on GitHub

1. Got to https://github.com/join?plan=free and follow the prompts.
2. Make sure you sign up for a **FREE** account, not a paid one!!! 😉
3. When you sign up and they ask for an email address, please use your UCSB.edu account (you can always add additional email addresses – like personal ones – to the same GitHub account later).

Part 2: Steps to getting your account ready on GitHub for CS16

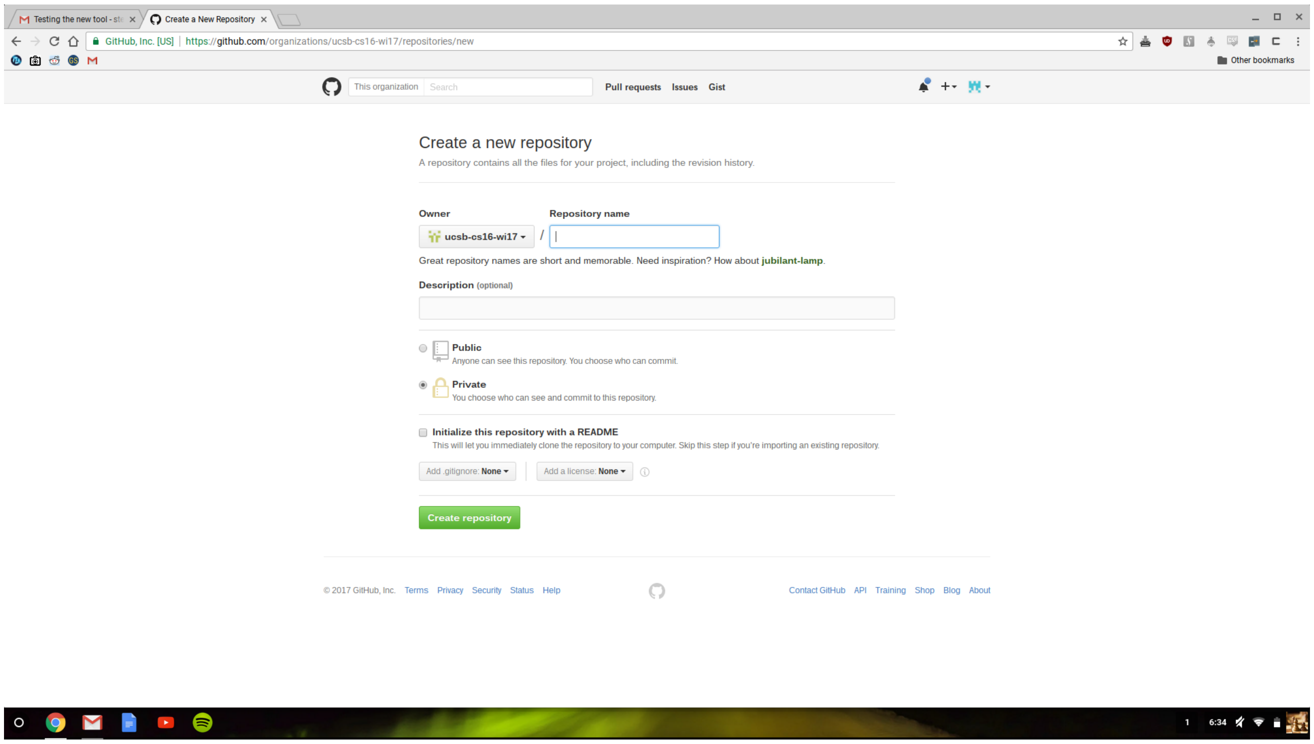
1. Make sure your UCSB.edu email is associated with your account (important step!). This means you might need to add it as a primary or backup email (for already existing accounts).
2. Go to this Google Form: **https://forms.gle/jaaVCrX4BouXHUJq9** and inform me about your GitHub account. I will then invite you as a member of our class' main GitHub "organization" (called **ucsb-cs16-f20-matni**). Look for that invite in your email and follow the prompts from there (you'll have to "accept" the invite).

Part 3: Creating a repo in our class' organization

For this lab and all subsequent programming assignments, you should start by creating a repo in the **ucsb-cs16-f20-matni** organization. Follow these steps:

1. Log into your GitHub account (if you haven't done so yet).
2. Navigate to your dashboard on the main GitHub website, i.e. https://github.com**.**
3. From the left drop down menu, select the class organization (this will only appear AFTER I've sent you an invite and you've accepted it).
4. Click on the green "New repository" button to create a new repository.
5. Type the name of your repo following the naming convention **lab02_your-github-username**. For example, if your GitHub username is **jgaucho**, you should name your repo **lab02_jgaucho**. This will create your repo inside our class' organization.

6. Select the "**Private**" visibility option so that other students in the org cannot view your code (but the prof. and TAs/ULAs, as "owners", will be able to view).
7. Add the **C++ .ignore** option from the drop down menu and click on "Create repository". See screenshot below.



Create a new repository

A repository contains all the files for your project, including the revision history.

Owner ucsb-cs16-wi17 / **Repository name**

Great repository names are short and memorable. Need inspiration? How about **jubilant-lamp**.

Description (optional)

( ) **Public**
Anyone can see this repository. You choose who can commit.

(●) **Private**
You choose who can see and commit to this repository.

[ ] **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository
© 2017 GitHub, Inc. Terms Privacy Security Status Help
Contact GitHub API Training Shop Blog About
6:34

8. Click on the "Create repo button".

Part 4: Uploading your Lab Files into Your GitHub Repo

Upload the files in your **lab02** directory to the new GitHub repo you just created. To do this, you would normally be physically present on a lab machine or in CSIL. However, in this online education environment, you will do this from your home computer. What you need to ensure FIRST, however, is that the **lab02** files to be uploaded (i.e. the 3 **.cpp** files) *are copied over on your home computer*.

1. On your web browser, navigate to your repo on GitHub. Click on the "Upload files" button.
2. Now either drag and drop the files: from your machine or use the "Choose your files" option to browse through your local directory and upload the file. Then press the green "Commit new files" button.
3. Navigate back to your repo to see that the files you uploaded are correctly listed. Click on it and you should see your code on GitHub's web interface.

Congratulations! You put your lab files on GitHub! 😊

You will be creating GitHub repos for every lab (each repo can hold multiple files). All these repos will be placed under the “umbrella” of our class’ GitHub “organization”.

Your GitHub repos will be useful as yet another way for you to review your programs with the professor, or the TAs, or the ULAs.

Step 5: Submit your Programs for Grading

Once you are satisfied that your programs are correct, then it’s time to submit them. While working with others is ok, you must STILL submit your OWN lab. Even if you’re working with another person, do NOT copy each other’s’ code!

Log into your account on **https://www.gradescope.com/** and navigate to our course site: **CS16 Fall 2020**. Select this assignment (**lab02**). Then click on the “Submit” button on the bottom right corner to make a submission. You will be given the option of uploading files from your local machine or submitting the code that is in your new GitHub repo. Choose either option and follow the steps to **upload ALL THREE (3) .cpp FILES** to Gradescope.