Due Date: 6 October, 2019 @ Midnight

Lab Objective

- Begin programming in C++.
- Exposure to Version Control (GitHub).
- Exposure to Vectors and ifstream. Introduction

C++ is a powerful, modern Object-Oriented Programming (OOP) language. In this lab, we will make the transition to programming in C++. We will utilize a data structure called a Vector in this lab. You likely haven't covered Vectors in lecture yet; however, we are going to utilize them in this lab due to the power of vectors.

In this lab, we will parse through a file that has 10,000 integers. You will store them into a vector and find the average of all the numbers.

The last portion of this lab will expose you to a version control system called Git. In this lab, we will utilize GitHub for our version-controlled repo; however, there are various other options available you may hear of. Many software engineering firms will ask you to share your GitHub link during interviews, so this is a great way to showcase your skills. I encourage you to continue to use GitHub outside of this lab, as knowing Git is an extremely valuable skillset and it prevents data loss.

Requirements:

- You must fill the vector using **emplace_back;**
- You must use a ranged for-loop (refer to lecture slides)
- You must open the file using ofstream.
- You must print the average to the terminal
- You must write a makefile for this lab.

Part 1: Opening a File in C++

In C, we utilized a File pointer (FILE *) to open files. In C++, we utilize the fstream library and create an object with either ifstream or ofstream. Ifstream should be utilized in this lab, as it will allow you to read from "numbers.txt".

Vectors

Vectors are dynamic arrays. Remember in C, where we either needed to know the size of the array or have some input for the array when we initialized it? There was no way to re-size the array without creating a separate array and copying values into it, but this is extremely difficult to do dynamically.

Vectors in C++ are dynamic arrays. Technically, they are a class, which means we have certain methods we can call on the vector. Here are a few:

- .emplace_back(<data here>)
    - This allows us to put as much data into the vector as we need. If it is full, it will resize automatically. You will learn about this in greater detail, but this is the method you want to call in this lab to store data inside of your vector.
- .reserve(<integer here>)
    - Sometimes we know how large our vector is, like in this lab. It is more efficient to go ahead and reserve the size of vector when the size is known.
- .size()
    - Remember in C how we had to keep a separate integer to remember the size of an array? The .size() method gives us an option to know the size of the vector at any time.

There are a lot more methods we can call; however, they are beyond the scope of this lab. Click here to learn more about vectors if you're interested.

Compiling C++ Programs

For the most part, you compile and run C++ programs similarly to C. Instead of gcc, we will use g++ to compile. When compiling in C++, I recommend adding the -Weffc++. It is utilizing Scott Meyer's techniques from *Effective Modern C++14*, which will ensure you programming utilizing best practices.

Example:

To compile:

- g++ <name_of_file>.cpp -Weffc++ -Wall -std=c++11

To run:

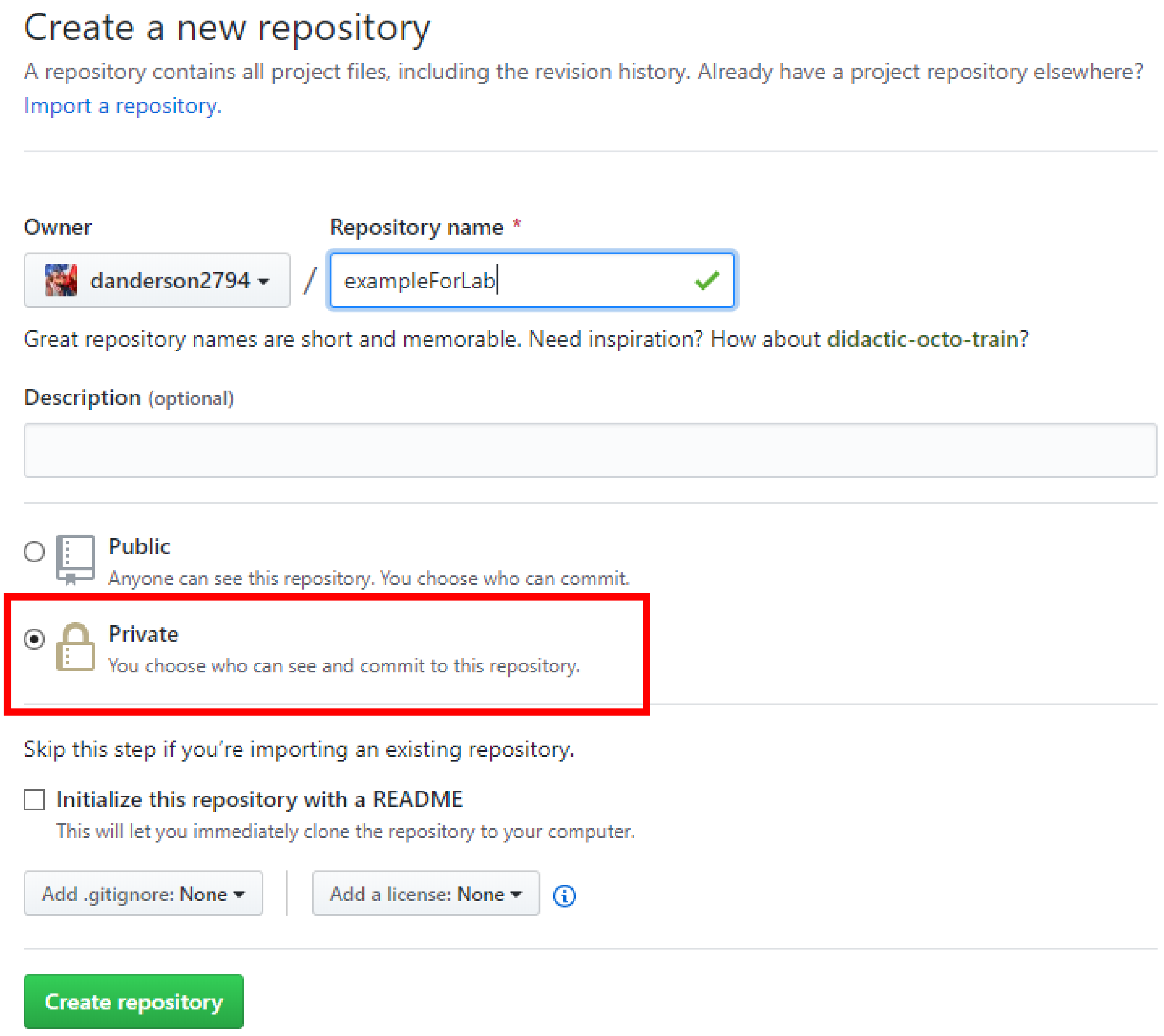
- ./a.out

GitHub

It is an industry standard to utilize GitHub, or some git system. Git is a version control tool developed by Linus Torvalds, the founder of Linux, in 2005. In a larger programming setting, it allows teams to checkout code, add features, and push it to the main program. It is a very powerful tool to manage software and teams.

In college, it is extremely useful to back up all of your programs and files. How many times have you made a change and wished you could revert to a previous version of a program? Or you're in the McAdams lab, but your files are on your personal laptop and sftp is a nightmare? Git allows us to bypass these issues.

Navigate over to https://education.github.com/students and **sign up with your Clemson.edu email.** This will give you access to expensive software for free while you're a student. Once you have created your account, create a **private** repo.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner: danderson2794 / Repository name *: exampleForLab

Great repository names are short and memorable. Need inspiration? How about didactic-octo-train?

Description (optional)

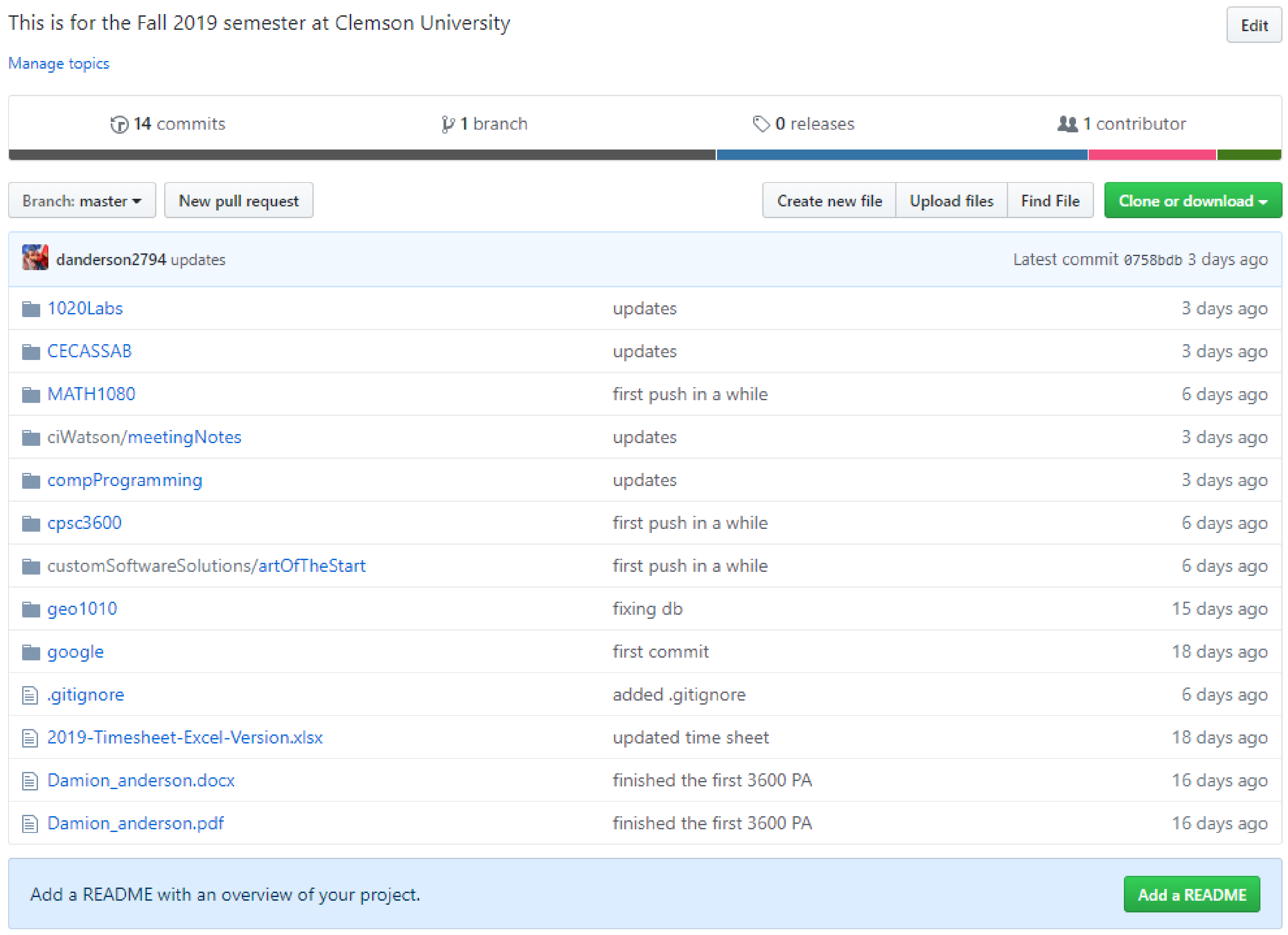
Public
Anyone can see this repository. You choose who can commit.

[✓ Private]
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Once you have created your repo, add your CPSC1020 files into your GitHub directory. I personally utilize GitHub for all of my classes, below is a screenshot of my Fall 2019 repo.



This is for the Fall 2019 semester at Clemson University

danderson2794 updates — Latest commit 0758bdb 3 days ago

1020Labs	updates	3 days ago
CECASSAB	updates	3 days ago
MATH1080	first push in a while	6 days ago
ciWatson/meetingNotes	updates	3 days ago
compProgramming	updates	3 days ago
cpsc3600	first push in a while	6 days ago
customSoftwareSolutions/artOfTheStart	first push in a while	6 days ago
geo1010	fixing db	15 days ago
google	first commit	18 days ago
.gitignore	added .gitignore	6 days ago
2019-Timesheet-Excel-Version.xlsx	updated time sheet	18 days ago
Damion_anderson.docx	finished the first 3600 PA	16 days ago
Damion_anderson.pdf	finished the first 3600 PA	16 days ago

Add a README with an overview of your project.

There is a GUI-based application you can use; however, I strongly recommend you learn the Git commands. Here are the most important ones to know:

- git init
    - Initializes the directory you're in and tells git you want to track this directory.
- git add <file names here>
    - This tells git to add these files to the repo and track any changes made to them
- git commit -m "your message here"
    - Commits are how we stage changes and files to be added to the repo. You always want to add a comment explaining what you're doing.
- git push
    - This will push the files / changes made in a commit to the repo.
- git pull
    - This will download your git repo and update your local directory.

When you create your repo, follow the directions on the page explaining how to add your files. Once you have uploaded your files, screenshot it and save it as a .jpeg with your username (<user_name>.jpeg).

What to Handin:

As with all labs, you must tar your files as <user_name>.tar.gz. Your tar file must include the following:

- Main.cpp
  - This must have the requirements mentioned above
- Makefile
  - It needs at least default, run, and clean commands
- <username>.jpeg screenshot of your GitHub account with your files uploaded.
  - **ENSURE YOUR REPO IS PRIVATE BEFORE YOU UPLOAD YOUR FILES, ELSE YOU MAY VIOLATE ACADEMIC HONESTY.**