Project #1

This is the final project for this class. The purpose of this project is for you to explore a topic we have covered in the lab in more detail. This will require you to do some research on your own; however, if you get stuck, please ask me for help. You should choose **ONE** of the following options, complete it, and put your results in the git repository for this project. Please make a note of which option you have chosen in the `README.md` file.

For each option, you must enter answers to questions in the `README.md`, as well as add any configuration files or scripts you wrote for the assignment to this repository. Your submission for this project should be similar to the lab assignments you have submitted so far.

Option A: Open source IDEs that support g++

For this option, you will become intimately familiar with a true IDE (as opposed to mere text-editor) that supports g++ (they must be open source, support g++, and ideally GDB and version control integration too). This means visual studio, visual studio code, and atom are not options for this assignment...

Examples include:

KDevelop, QtCreator, Code::Blocks, Codelite, Geany, Eclipse CDT, Netbeans with c++, Anjuta, etc.

Note: if you want to turn Vim or Emacs into an IDE with plugins and extra configuration to do this project, you can document that process.

You may want/need to install Linux in a VM to try some of these: for a guide, see:
https://www.cnsr.dev/index_files/Classes/DataStructuresLab/Content/00-VirtualMachines.html

1. Learn and test the debugging features of your chosen IDE. Document the details with screenshots and text narrative.
2. If it has version control, learn and test the version control integration features of your chosen IDE. Document with text narrative and screenshots uploaded to your repository.
3. Does the IDE support build systems like make, cmake, makedepend? If so, test and describe briefly how to use them with the IDE.
4. Explore and describe 10 new IDE features.
5. Does it have plugin capability? If so, try out 2 plugins, and document.

Option B: Shell Scripting

For this option, you will discover some fancy shell scripting features.

1. Read both of the following in-full:
   http://guide.bash.academy/
   http://www.linuxcommand.org/lc3_learning_the_shell.php

2. What is command substitution? Write at least two example shell scripts that use it. (Hint: `bc` will do non-integer math.)

3. Read the job control and processes slides from the basic bash day.
   a. What happens to paused or background jobs when you log out?
   b. What does the command `disown` do?
   c. What does the command `wait` do? Write a shell script that uses `wait` on at least two jobs (you can use `sleep` to make processes that run for a while).

4. Try and document 10 new commands or tasks you did not already know.

Option C: Version Control

For this option, you will get to experiment with some more complicated git features.

1. Read as much of https://git-scm.com/book/en/v2 as you can
2. `git rebase` (Hint: make a test repository to experiment with! Don't try it out in the repository for this project.)
   a. Make two branches with commits on each. Instead of merging one branch into another, rebase it instead.
      What does the commit graph look like, and how does it differ from a merge?
   b. Check `git log` before and after a rebase. What changes about the commits you have rebased?
   c. Perform an interactive rebase (`-i`). What things can you choose to do with each commit? Try some of them out (and document what you did).
3. What does `git stash` do?
4. How can you use `git checkout` to get the contents of a file from a specific commit?
5. Try and document at least 5 new features you did not already know.

Option D: Alternate build systems

For this option, pick one of the following tools and use it to automatically generate as much of Lab 5 Problem 2 as you can. Describe what the tool does, and how to use it.

1. https://en.wikipedia.org/wiki/Makedepend
2. https://cmake.org
   https://en.wikipedia.org/wiki/CMake

CMake may be especially interesting if you enjoy using IDEs since it can generate project files for several common IDEs in addition to makefiles.

Option E: Debuggers

For this option, try out all the debugging front-ends (gdb, Kdevelop, qtcreator, codeblocks, kdbg, and cgdb) to accomplish the following:

1. Launch the debugger and use: start, step, next, finish, print, set, etc.
2. Interact with the gdb command line in the front-end (kdgb excluded)
3. Determine how to launch with program arguments
4. Determine how to launch with standard input redirected from a file

Include screenshots of the configuration needed for gdb command interaction, as well as for arguments and redirection, including the successful result. In the readme you include, describe the pros and cons of each, and which you like the best.