What's up brahs?

Aiden Cox - CEG 2350 Lab Lead
Austin Kellough - CEG 2350 Lab Assistant
Quote of the week:
"Everyday might not be a good day but there's something good in everyday"

How was Midterm and Lab09?

Processes and system-y things?
Midterm questions?

Second AWS Check-In!

1. Ability to connect to AWS instance
2. Command history relevant to Lab 09 and / or 10
3. At least one branch in GitHub repository

```
                        ubuntu@ceg2350-sandbox: ~
1923  git switch main
1924  git pull
1925  git branch
1926  git pull
1930  git switch sigma
1932  git switch main
1934  git branch
1935  git merge sigma
1937  man git
1938  man git add
1939  man git branch
1940  man git-branch
1941  git checkout -b 67
1942  git branch
1944  git switch
1945  git switch main
1946  git pull
1948  git push
1949  git branch
1950  git logs
1951  git log
1953  git branch
1960  git clone git@github.com:aidenc17/CEG3310.git
1966  git pull
```

Check off in-person (with your instructor or TAs)

Beginning Lab10

Lab Instructions: https://pattonsgirl.github.io/CEG2350/Labs/Lab10/Instructions

Lab Template:
https://raw.githubusercontent.com/pattonsgirl/CEG2350/refs/heads/main/docs/Labs/Lab10/LabTemplate.md

Feeling comfortable with the course?
Questions from lecture?

Issues and Branches - Part 0

Part 0 - Issues & Branches

Useful `git` commands: `branch, switch, commit, push, merge`

1. To begin, create and add the Lab Template to your `main` branch in your `Lab10` folder
2. In your GitHub repository, go to the `Issues` tab. Create an Issue **for each part** of this lab - you may do this all at once or as you start each part.
3. Through this lab, you will create **one** branch **for each** issue.
4. Merge the branch with the `main` branch as your **complete each part**. You may then mark your issue as complete.
5. If you make **adjustments after merging** you need to create an Issue describing what needs correcting, then resolve the issue.

One merge must be completed via a **Pull Request** - all others should be done via branch merging on the command line.

Make sure that EACH PART of lab has an issue, a branch, and that each one gets MERGED via command line, or a Pull Request.

My Girlfriend Says I have a lot of Issues? - Part 0

pattonsgirl / CEG2350

<> Code   ⊙ Issues 17   ⇅ Pull requests   ⊙ Actions   ⊞ Projects   📖 Wiki   ⊘ Security   📈 Insights   ⚙ Settings

is:issue state:open

🏷 Labels   ⬦ Milestones   **New issue**
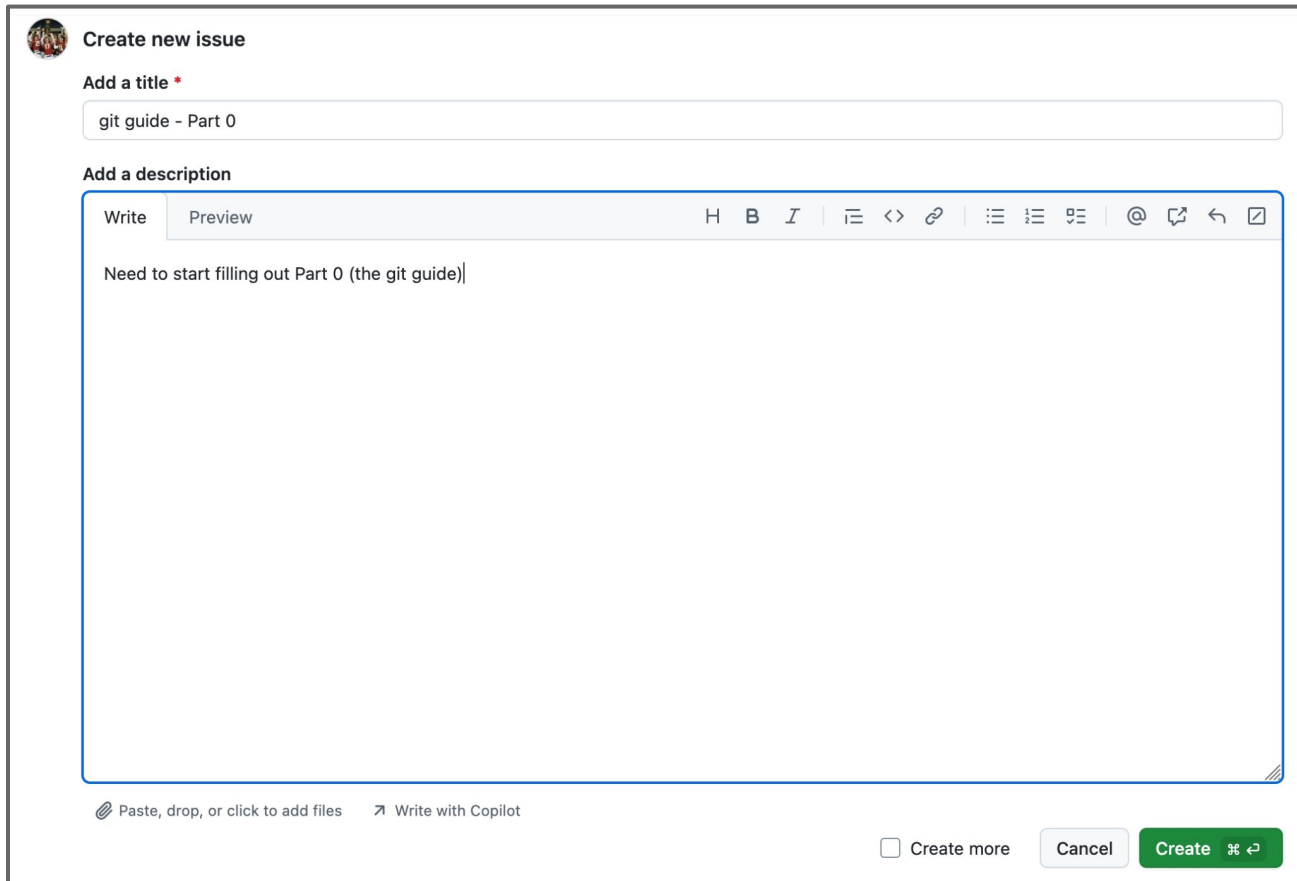
Open 17   Closed 14

Author ▾   Labels ▾   Projects ▾   Milestones ▾   Assignees ▾   ⬇ Newest ▾

⊙ **Lab 08 - unique instance?**
#54 · pattonsgirl opened 2 weeks ago

⊙ **Lab 04 - Add Main to bash?**
#47 · aidenc17 opened on Sep 26

⊙ **lab 02 clarification / wording issue**
#45 · b-wens-kedar opened on Sep 11

⊙ **Grading & GH Issues**
#43 · pattonsgirl opened on Sep 5

⊙ **Lab 12 - Citations used for Part 5 (need some bold letters)**
#41 · aidenc17 opened on Apr 26

⊙ **Lab 11 - Extract and Profit Permissions**
#40 · aidenc17 opened on Apr 21

⊙ **Exam 2 Missing**
#38 · pattonsgirl opened on Apr 4

⊙ **Lab 06 - relative pathing or use ENV variable**
#35 · pattonsgirl opened on Mar 11

Our course Repo's Issues!

My Girlfriend Says I have a lot of Issues? - Part 0

Create new issue

Add a title *

git guide - Part 0

Add a description

| Write | Preview |

Need to start filling out Part 0 (the git guide)

🔗 Paste, drop, or click to add files   ↗ Write with Copilot

☐ Create more   Cancel   Create ⌘ ↵

You will end up making an issue for each part of the lab!

The button for this is found under issue, and new issue on the top right.

After creating the issue, the next slide shows what it looks like to click back into it.

My Girlfriend Says I a have lot of Issues? - Part 0

git guide - Part 0 #2

Issue title

Open

aidenc17 opened 22 minutes ago

Need to start filling out Part 0 (the git guide)

Create sub-issue

Issue description

Add a comment

Write Preview

Use Markdown to format your comment

Paste, drop, or click to add files

Button to close issues

Close issue Comment

Edit New issue

No one - Assign yourself

Labels
No labels

Type
No type

Projects
No projects

Milestone
No milestone

Relationships
None yet

My Girlfriend Says I have a lot of Issues? - Part 0
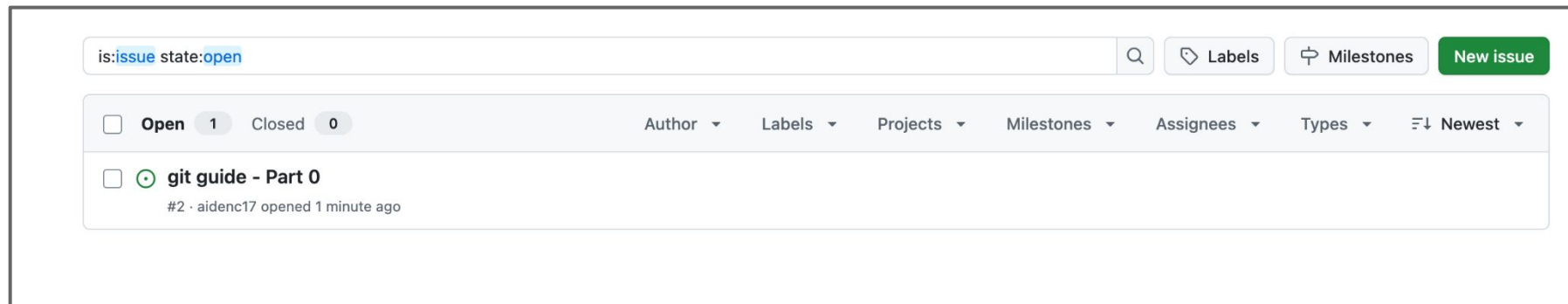
After clicking create, you'll go to the main issue page. Your's will all be populated with your grades for the labs, but you can catch my drift!

My Girlfriend Says I have a lot of Issues? - Part 0

WSU-kduncan / **ceg2350f25-aidenc17** ⑂

Type / to search

<> Code   ⊙ **Issues** 6   ⑂ Pull requests   ⊙ Actions   ⊞ Projects   📖 Wiki   ⓘ Security   📈 Insights   ⚙ Settings

is:issue state:open

🏷 Labels   🏁 Milestones   **New issue**

☐ **Open** 6   **Closed** 0

Author ▾   Labels ▾   Projects ▾   Milestones ▾   Assignees ▾   Types ▾   ⇅ Newest ▾

☐ ⊙ **makey makey - Part 5**
#7 · aidenc17 opened 2 minutes ago

☐ ⊙ **document - Part 4**
#6 · aidenc17 opened 2 minutes ago

☐ ⊙ **ignore that - Part 3**
#5 · aidenc17 opened 3 minutes ago

What my repo looks like after creating all the issues

☐ ⊙ **compiling - Part 2**
#4 · aidenc17 opened 5 minutes ago

☐ ⊙ **common compilers - Part 1**
#3 · aidenc17 opened 7 minutes ago

☐ ⊙ **git guide - Part 0**
#2 · aidenc17 opened 38 minutes ago

Branches (But not tree ones. Well, kinda) - Part 0

On the course repo, make a branch!
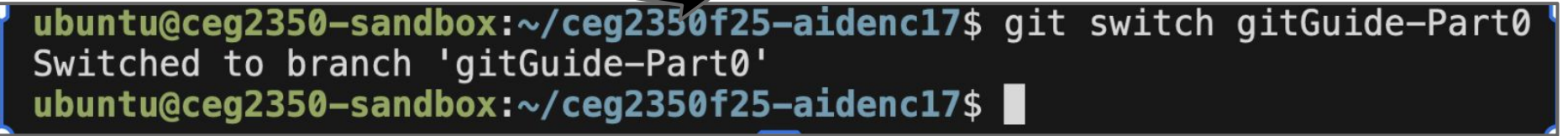
ubuntu@ceg2350-sandbox: ~/ceg2350f25-aidenc17

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17$ git branch gitGuide-Part0
```

Swap to new branch!

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17$ git switch gitGuide-Part0
Switched to branch 'gitGuide-Part0'
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17$
```

Do this for each of your issues, (all of the parts) and commit to each of those branches. When complete, merge branches. One MUST be done using a Pull Request (PR)

Branches (But not tree ones. Well, kinda) - Part 0

When trying to push from a new branch, you will get an error message!

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17$ git push
fatal: The current branch compiling-Part2 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin compiling-Part2

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

That's expected! Just run the command it tells you to and life will be good!

Branches (But not tree ones. Well, kinda) - Part 0

What my repo looks like after the branches are made:


```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17$ git branch
  commonCompilers-Part1
  compiling-Part2
  document-Part4
  gitGuide-Part0
  ignoreThat-Part3
* main
  makeyMakey-Part5
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17$
```

Inside my AWS instance / Command Line

GitHub Website

⑂ main ▾   ⑂ 7 Branches   🏷 0 Tags

Switch branches/tags   ✕

🔍 Find or create a branch...

Branches   Tags

✓ main   default

commonCompilers-Part1

compiling-Part2

document-Part4

gitGuide-Part0

ignoreThat-Part3

makeyMakey-Part5

View all branches
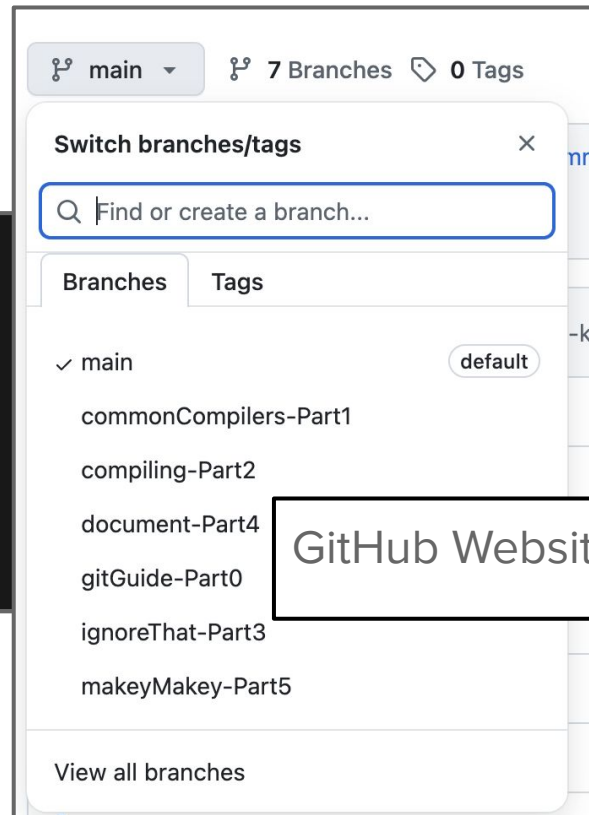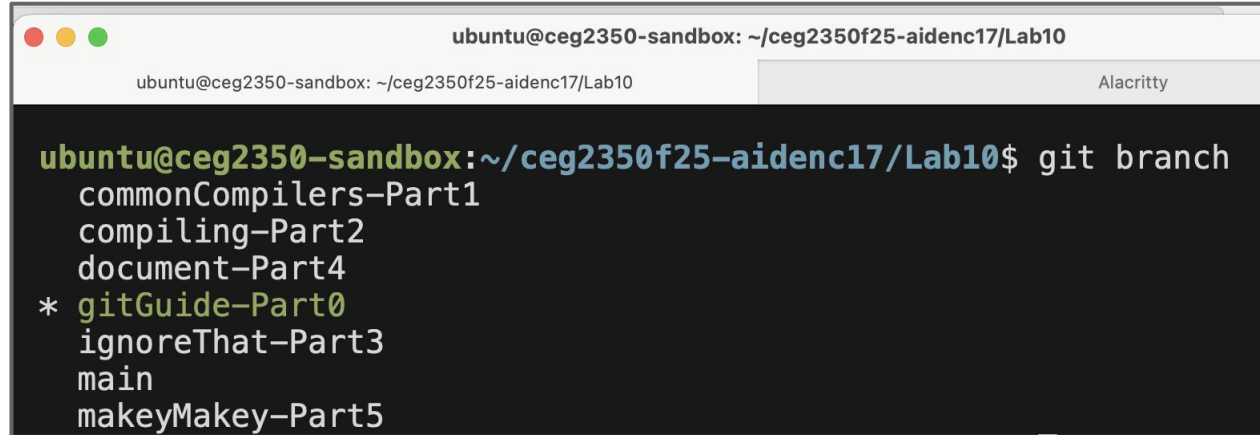
Merging (But Not as Bad as Merge Sort) - Part 0

After making changes to a branch, the changes only pertain to that branch. Swapping to the Main branch will contain the changes that were made before branching.

To bring the changes over from our Part0 branch, (and the rest of the branches for this lab), you will need to do a `git merge`.


```
ubuntu@ceg2350-sandbox: ~/ceg2350f25-aidenc17/Lab10
ubuntu@ceg2350-sandbox: ~/ceg2350f25-aidenc17/Lab10          Alacritty

ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ git branch
    commonCompilers-Part1
    compiling-Part2
    document-Part4
  * gitGuide-Part0
    ignoreThat-Part3
    main
    makeyMakey-Part5
```
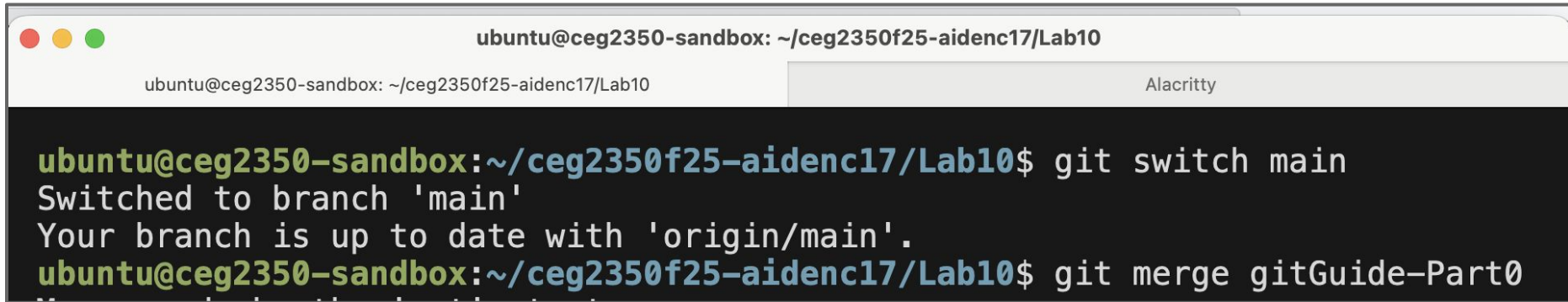
One of them will be a Pull Request, more on that later

We want to be on the branch that we are merging INTO. Swap on over to Main!

Merging (But Not as Bad as Merge Sort) - Part 0

ubuntu@ceg2350-sandbox: ~/ceg2350f25-aidenc17/Lab10

ubuntu@ceg2350-sandbox: ~/ceg2350f25-aidenc17/Lab10   Alacritty

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ git merge gitGuide-Part0
```

Congrats! You have successfully merged (moved) your changes from the Part0 branch into Main!

Next on how to do a Pull Request

Pull That Shit In - Part 0

After making changes to a branch, you may see the big scary green button on GitHubs website for your course repo. You will need to use this button for ONE of your merges.

Pull That Shit In - Part 0

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also **compare across forks** or **learn more a**

base: main ⌄ ← compare: gitGuide-Part0 ⌄ ✓ **Able to merge.** These branches can be automatically merged.

Add a title

Notice we are going to now "merge" or bring our changes from our Part0 branch over to our main branch.

H B I ≣

...merged in via accepting this PR

"Able to merge" means there aren't any merge conflicts going to arise. Perfect!

Ⓜ Markdown is supported   🖼 Paste, drop, or click to add files

Create pull request ⌄

Pull That Shit In - Part 0

PR is now under the PR tab, and upon clicking the "merge pull request button", main will have all of the changes that our branch had!

Compilery Things - Part 1

Part 1 - Compilers

Reminder - create an Issue for this part -> create a branch to work on this issue -> make your commits to this branch -> verify your changes (no broken markdown, good spelling) -> merge to main

Fill out the following charts regarding the three most commonly used languages - C/C++, Java, and Python. The bonus is you'll make a guide for yourself on how to install these or how to check if they are already available. I'm going to give you the answers for C / C++ - you'll need to complete the chart for Java and Python.

Fill out the tables with information on where to install the languages and "how to verify" - which / whereis and version (--version) on each Linux, Mac, and Windows.

Compilery Things - Part 1


```
ubuntu@ceg2350-sandbox:~$ gcc --version
gcc (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0
Copyright (C) 2023 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  T
here is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICUL
AR PURPOSE.

ubuntu@ceg2350-sandbox:~$ python3 --version
Python 3.12.3
ubuntu@ceg2350-sandbox:~$ javac --version
javac 21.0.8
ubuntu@ceg2350-sandbox:~$ 
```

Your AWS instance is Linux!

Compiling on the Command Line - Part 2

Part 2 - Compiling

Reminder - create an Issue for this part -> create a branch to work on this issue -> make your commit to this branch -> verify your changes (no broken markdown, good spelling) -> merge to main

Useful Commands: `gcc, g++, wget`
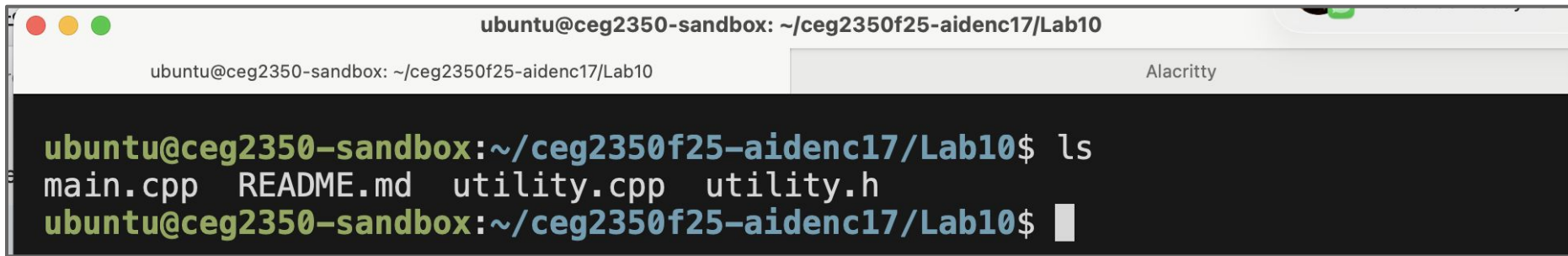
1. Select from either TODO-C or TODO-CPP. Download the files in the folder to the `Lab10` folder in your repository folder. `add`, `commit` and `push` the source code files.
2. Install the C / C++ compiler on your AWS instance.
3. Verify the installation of the C / C++ compiler on your AWS instance.
   - It is recommended to query for the version of the program to validate installation.

4. Using your AWS instance, compile the source code into an executable program.
5. Using your AWS instance, run the executable program.
6. Add the program source code file - `.c` or `.cpp` files - to your repository. Do not add the compiled executable for tracking.

Grab either the `c` or `c++` code from the course GitHub repo, and figure out a way to pull them onto your system!

Follow Part 1 on how to install compilers and verification of them.

Next slide for compilation process!

Compiling on the Command Line - Part 2

Grab either the `c` or `c++` code from the course GitHub repo, and figure out a way to pull them onto your system!

Follow Part 1 on how to install compilers and verification of them.

Next slide for compilation process!

Compiling on the Command Line - Part 2


```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ g++ main.cpp utility.cpp utility.h -o aidensProgram
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ ls
aidensProgram   main.cpp   README.md   utility.cpp   utility.h
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$
```

Since I snagged the `c++` files, I used `g++` to compile. The `-o` specifies the output file, and the ARG after give the program the name.

Run the program like you would an executable with the executable bit!

.gitignore Me Like My Girlfriend Does - Part 3

Put a file named `.gitignore` in the root of your repository! (Outside all of the Lab folders)

Inside the file, list files you do not want to track. This way, they aren't added to GitHub, and when using shortcuts git will completely skip over it

Document the Code - Part 4

Example of best practice formatting - do this for each function in your `.c` or `.cpp` files:

```
/*
 * Function:  approx_pi
 * --------------------
 * computes an approximation of pi using:
 *    pi/6 = 1/2 + (1/2 x 3/4) 1/5 (1/2)^3  + (1/2 x 3/4 x 5/6) 1/7 (1/2)^5
 *
 *  n: number of terms in the series to sum
 *
 *  returns: the approximate value of pi obtained by suming the first n term
 *           in the above series
 *           returns zero on error (if n is non-positive)
 */

double approx_pi(int n) {
  ...
}
```

```cpp
#include "utility.h"

TodoList::TodoList() {}



// this is where the commetn should go
// what func does
// params of func
// what it returns
// example JAVA code provided if you are unsure what the code does!
void TodoList::displayMenu() {
    std::cout << "\n===== To-Do List Menu =====" << std::endl;
    std::cout << "1. Add Task" << std::endl;
    std::cout << "2. Mark Task as Completed" << std::endl;
    std::cout << "3. Remove Task" << std::endl;
    std::cout << "4. View To-Do List" << std::endl;
    std::cout << "5. Exit" << std::endl;
}
```

Using the code from Part 2, go over all of the files and make a comment block on each function including what the function does, parameters, and returns.

Makey Makey - Part 5

Tired of using Part 2's code? Too bad. Let's create a `Makefile` to build, run, and clean up our TaskToDo program.

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ touch Makefile
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$ ls
main.cpp  Makefile  README.md  utility.cpp  utility.h
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/Lab10$
```

A `Makefile` is just a text file that describes how to build your system. In this case, how to build our program so we don't have to type out the long command each time

Makey Makey - Part 5

Tired of using Part 2's code? Too bad. For this part, create a `Makefile` to build, run, and clean up our TaskToDo program.
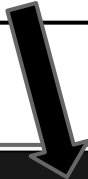
Demo directory with example cpp files!

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ ls
file2.cpp   header.h   main.cpp   Makefile
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$
```

Makey Makey - Part 5

Example Makefile:

These are targets (what you put after `make` is called)

Dependencies (what is needed to do the things

```
aiden: main.cpp file2.cpp header.h
        g++ -Wall main.cpp file2.cpp -o aiden

execute: aiden
        ./aiden

remove:
        rm -f aiden
~
```

Commands. The "do things part"

Makey Makey - Part 5

Running the targets:

```
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ ls
file2.cpp  header.h  main.cpp  Makefile
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ make aiden
g++ -Wall main.cpp file2.cpp -o aiden
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ ls
aiden  file2.cpp  header.h  main.cpp  Makefile
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ make execute
./aiden

===== To-Do List Menu =====
1. Add Task
2. Mark Task as Completed
3. Remove Task
4. View To-Do List
5. Exit
Enter your choice (1-5): 5
Exiting program. Goodbye!
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ ls
aiden  file2.cpp  header.h  main.cpp  Makefile
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ make remove
rm -f aiden
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ ls
file2.cpp  header.h  main.cpp  Makefile
ubuntu@ceg2350-sandbox:~/ceg2350f25-aidenc17/demoMakeFile$ 
```

Have a Good Weekend!

Don't hesitate to reach out and ask questions!
Quote of the week:
"Everyday might not be a good day but there's something good in everyday"