Table of Contents

[GitHub Classroom Instructions 2](#__RefHeading___Toc181_1137193301)

[Instructor Instructions 2](#__RefHeading___Toc183_1137193301)

[GitHub Classroom Setup 2](#__RefHeading___Toc185_1137193301)

[Downloading the Class Roster 4](#__RefHeading___Toc190_1137193301)

[Student Instructions 5](#__RefHeading___Toc187_1137193301)

[Aces 5](#__RefHeading___Toc192_1137193301)

[Student Use 5](#__RefHeading___Toc194_1137193301)

[Instructor Use 6](#__RefHeading___Toc196_1137193301)

[Code Explanation 6](#__RefHeading___Toc207_2896572452)

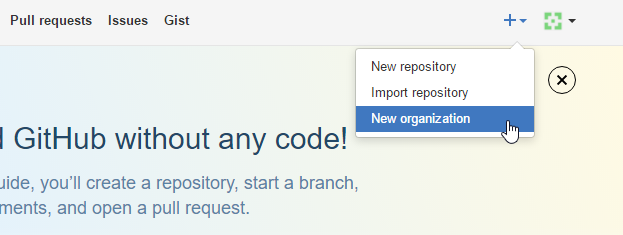
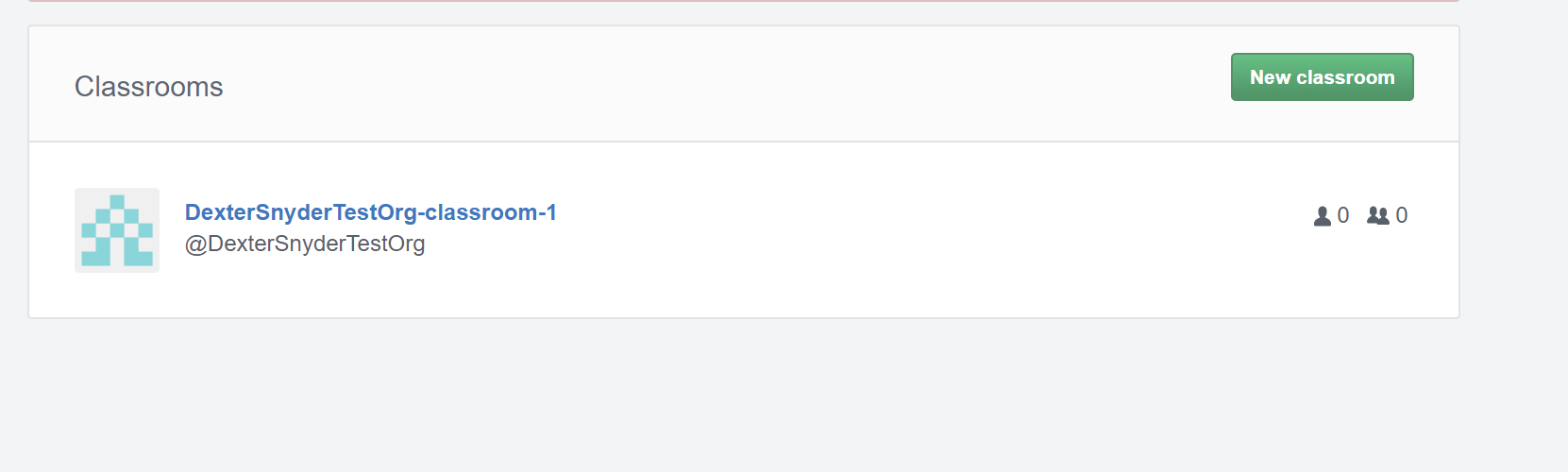
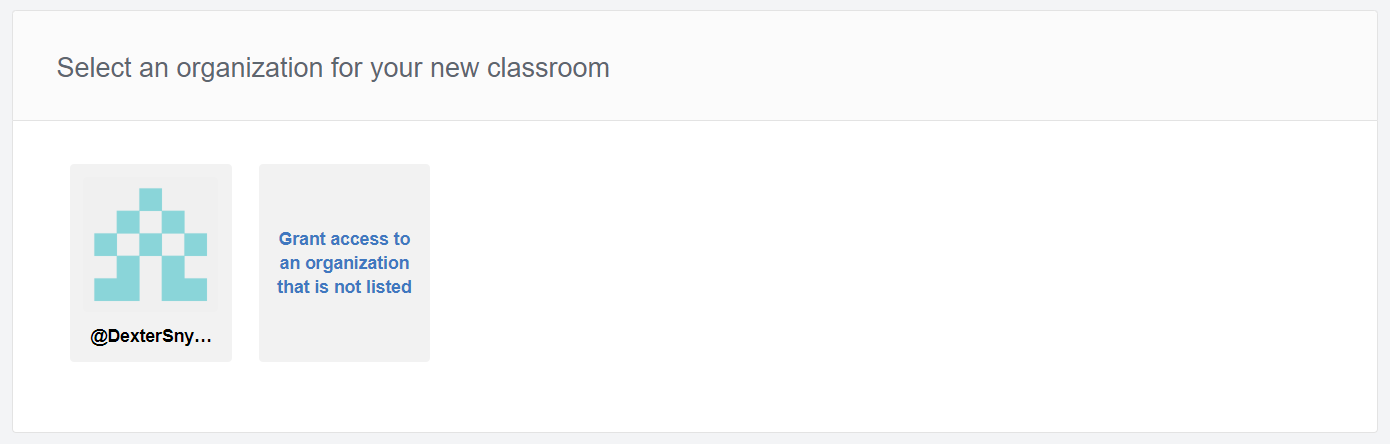
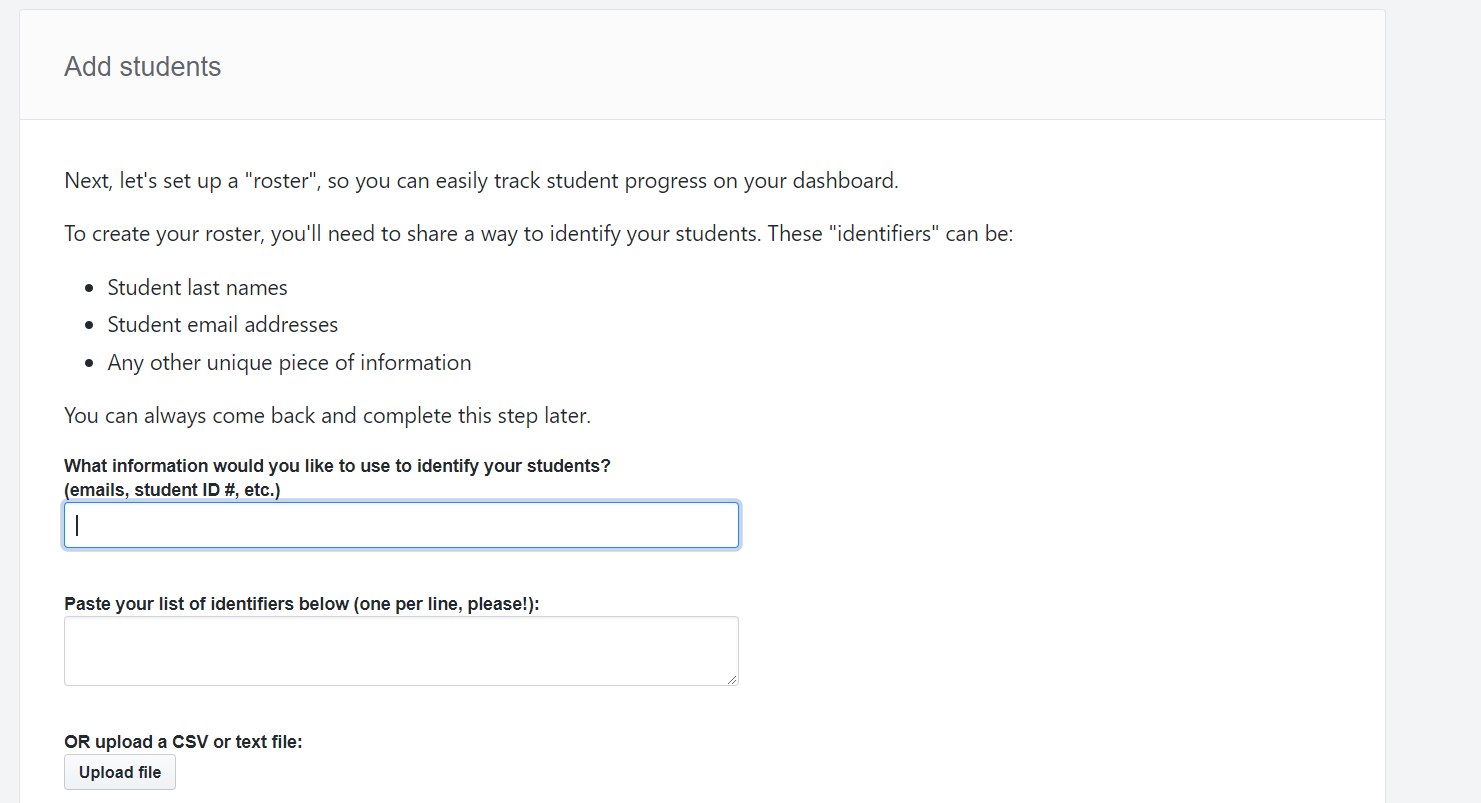
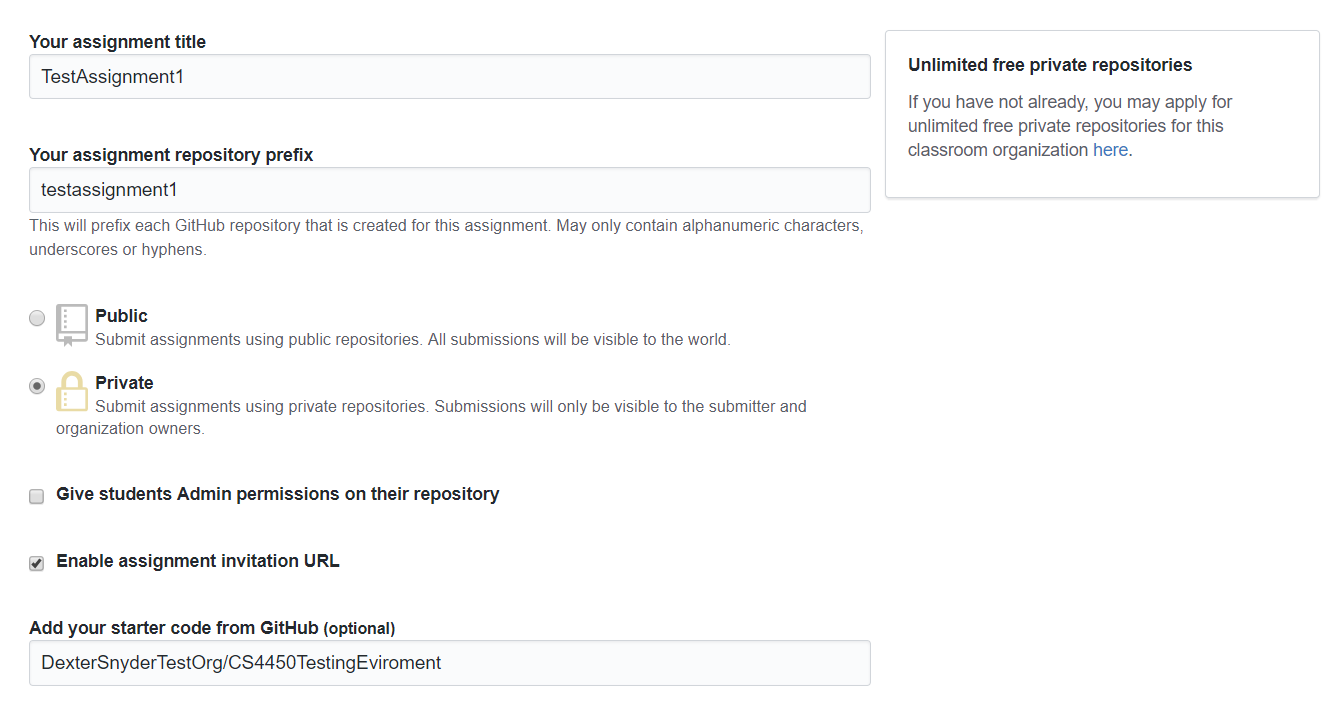
[Unit Test Files 6](#__RefHeading___Toc219_1400621271)

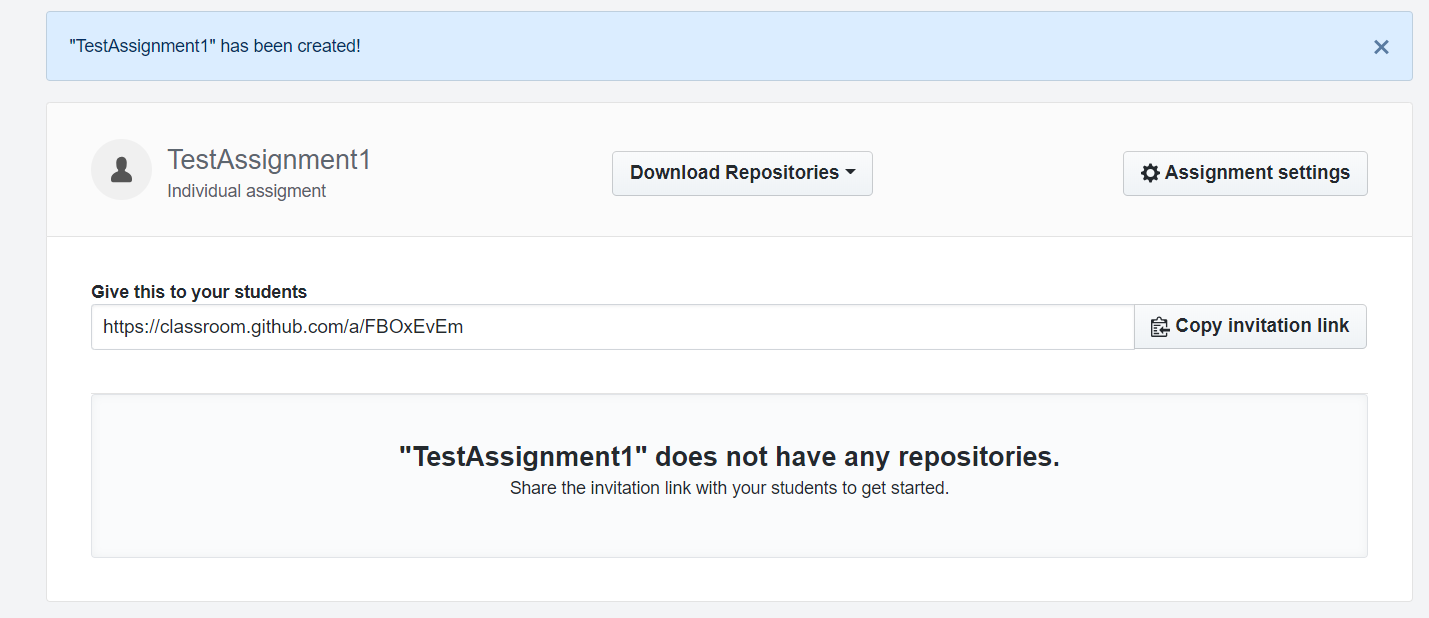
[To do 7](#__RefHeading___Toc198_1137193301)

# GitHub Classroom Instructions

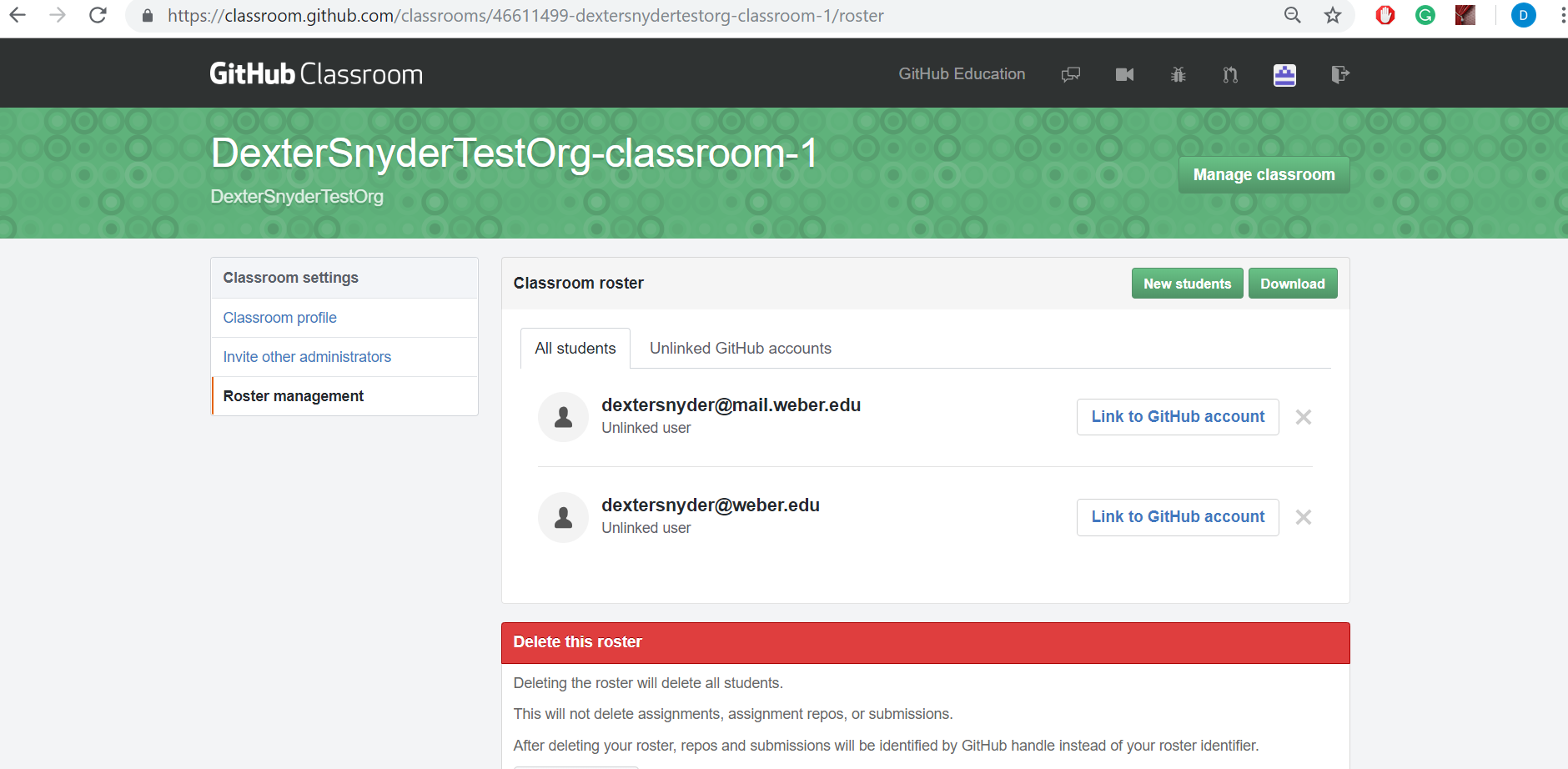
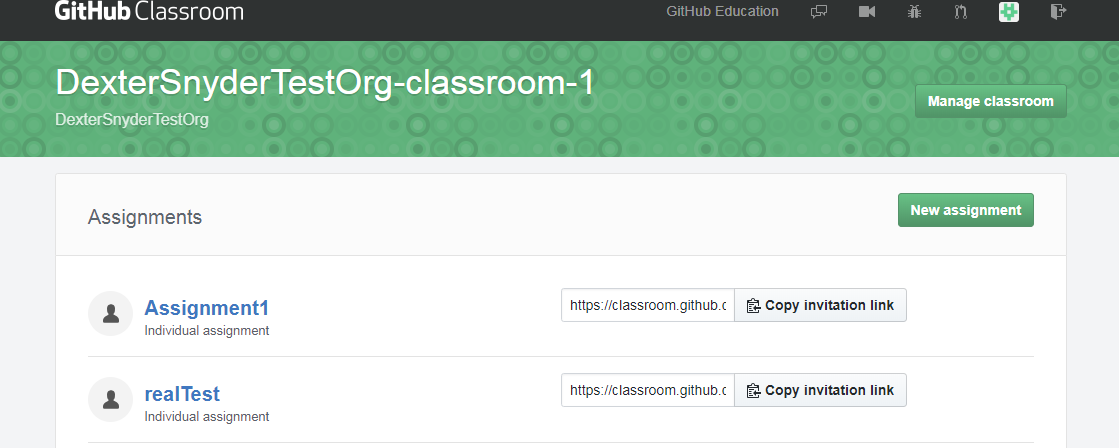
## Instructor Instructions

### GitHub Classroom Setup

1. Go to github.com and sign in
2. Create a new organization
3. Enter the information and choose the free plan (note that all classrooms you create will be under the organization name you choose)
4. Create a new repository for the starter code of the assignment you what to add. (Make sure to add \*.opendb to the gitignore file, or Visual Studio will break the auto commit)
5. Go to classroom.github.com, sign in, and authorize github
6. Create a new classroom
7. Select the organization you created
8. Choose the identifiers students will use to identify themselves when linking their github account.
9. Create an assignment
10. Click create an individual assignment
    1. Enter an assignment name & assignment repository prefix. (It is recommended to make these exactly the same, because in the Aces program when it asks for the assignment name it is really asking for the assignment repository prefix.)
    2. Add the starter code repository you created in step 4
11. Now that you have created an assignment it will show you the URL to give to the students



### Downloading the Class Roster

1. Go to manage classroom.
2. Go to roster management then download roster.

## Student Instructions

Students must have or create a GitHub account.

* Students can create an account by following the instructions below:
  + First go to github.com
  + Next, on the top right corner of the page, click on Sign Up or they can click on Sign Up for GitHub in the center of the homepage
  + Then, create a username and password
  + Finally, after that, choose the free account option for the subscription
* This must be done in order for students to submit their assignments and receive a grade for their work.
* Once students’ accounts are set up, the professor will email the students an invitation to join their GitHub Classroom.
  + Students will click on the link which will take them to a GitHub Classroom page where they will locate their name and add themselves to the classroom.
  + Once they are added to the classroom the students will then be able to clone the repos to their assignments when the professor sends out the links to clone them.
* Students will then clone the repository, and complete the work on their local machines.
* In order to submit the assignment, the student will complete the following steps
  + Git add
  + Git commit
  + Git push

# Aces

## Student Use

## Instructor Use

## Code Explanation

# Unit Test Files

For every assignment there must be two unit tests the instructor version that is used when running Aces, and a student unit test.

ACES\_Cpp contains a cpp function that will automatically do a git commit each time it is run. In order to work, this must be run each time the student runs the program. The best way to do this is to include it in a main method that the student doesn’t modify.

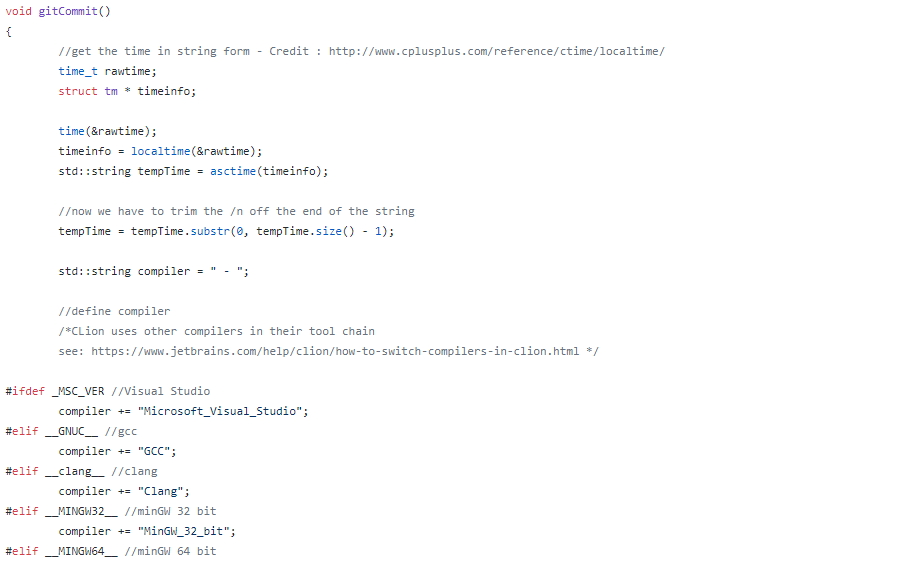
There are 3 files that should be in a student assignment:

* .Cpp file where the students do their work
* .H file (optional) for the students to work in
* Unit test file. This should contain the main() method, the unit test for the program, and the git commit function

In addition to the files given to the students, the instructor will need a unit test file with several modifications

* It should not contain the git commit function
* A successful unit test should output the string “Passed [security code]” where the security code is a user defined string
* It should contain a main() method

When ACES is run, the student unit test is deleted and the instructor version is copied in its place.



# To do

* Save assignment data in .net core Session data
  + Currently the assignment data is stored in a static variable inside the AssiginmentServices Class. This should be deleted and the data should instead be stored in session data, so that the program can handle multiple instructors using it concurrently. This should also fix the problem of not being able to click the back button on the assignment details page.
* Validate input
  + The program only validates the login. It should check that all other input (assignment name, security key, etc.) is valid
* Secure login
  + Use OAuth to login with a token or find another way to login without using the instructors plain text password to login.
* Handle problems with student's code
  + Aces needs to be able to handle any problems with the students code that would prevent the program from finishing (e.g. infinite loops).
* Compile and run student's code async
  + To make the program more efficient, and so it does not wait for too long waiting on one students code that has problems.
* Permissions for student uploads
  + Anything the students upload should not have the ability to damage or modify the server, or anything on it.
* Make it so students cant access the Professor Screen
  + The students need the URL to access the Professor screen, but if the get that URL they can add & delete classes. Maybe add a login page to prevent students from getting access to it.