CSCI 235

Software Design & Analysis II

Hunter College - Fall 2022

Programming Guidelines:

It is **extremely important** to follow these guidelines. **Failure to follow them will result in a lower grade or no credit at all for your assignment**. Read the following carefully!

These are general submission instructions. You should also follow closely the instructions on each project specification for details specific to individual programming projects.

All programming projects must be submitted on **Gradescope (via Github Classroom, details in project specification)** no later than the due date. You have been sent email invitations to Gradescope. Make sure you **login to your Gradescope account right away**. If you have problems logging into Gradescope, submit a request on Ed Discussion (comment to related post there) or seek help from the UTAs immediately during tutoring sessions.

Submit only the requested source files (.hpp and .cpp). Do not submit executables.

All programming projects must be submitted by **11pm on the due date**.

No credit will be given for assignments submitted late. You may however submit multiple times before the due date and only the last submission will be graded. So **be proactive** and **submit early**, this will help you find out if your project has problems and still give you time to fix it.

Although Gradescope allows multiple submissions, **it is not a platform for testing and/or debugging and it should not be used for that. You MUST test and debug your program locally.**

While you are encouraged to discuss project assignments with others, **all work submitted must be your own.** You MAY NOT show your solution to a classmate or ask another student to see their solution. You may not ask another student or UTA to debug your code. You may not use code from the Internet (e.g. StackOverflow). Sometimes it may be appropriate to use a small snippet of code from your textbook or other official source. To do so you must first ask the instructor to confirm it is appropriate, and you must do so with attribution (add a comment citing in detail the source of the code — you must always do this whenever you find yourself using others' code). You may not post your code where it is accessible to others, and you may not seek help from online forums. As a rule of thumb, **you must type and debug your code without directly copying someone else's code**. For the first incident of cheating or plagiarism your grade will be a 0 and it will not be dropped as the lowest. For the second incident, you will fail the class. We report all incidents to the Office of Student Affairs.

Every program must be professionally documented. Every distinct source code file must contain a preamble with the file's title, author, brief description, date of creation. All functions must have a prologue containing comments for each parameter, where appropriate pre and post conditions and return values. You should strive for self-commenting code. However, all nontrivial algorithms must be documented in plain English in a multiline comment block. All nontrivial declarations must have adjoining, brief comments. **Proper documentation is worth 10% of your project's grade (unless differently specified the project description).**

Please note that **all programming project submissions must compile and run with g++ without issue on the Linux lab machines** on the 10th floor of Hunter North. These computers provide a common platform to evaluate program execution, free of issues related to OS or IDE. You should always confirm that your assignment code successfully compiles and executes on these machines before submitting to Gradescope. "But it ran on my machine!" is not a valid excuse for a submission that does not compile. You have all been given accounts on these machines. If you receive an email about a new linux account, follow instructions. If you already have an account you will not receive an email and **must reclaim your account within two weeks of the beginning of class** by typing `touch fall.2022` in terminal after logging in. For more instructions follow this link: http://compsci.hunter.cuny.edu/~csdir/

If you are working on Windows, you may follow instructions on the "**Installing Windows Subsystem for Linux**" tutorial here: https://okunhardt.github.io/documents/Installing_WSL.pdf. If you are working on Mac you can access the terminal in Applications/Utilities. If you are working in Linux a terminal is also available.

You can remotely login to the lab machines as follows (in a terminal window do the following):

1. To upload your programs on one of the cslab machines you can use **sftp** in order to transfer your code to eniac. First navigate to the local directory where the files you want to upload are. Then to sftp to the sever type:
   `sftp your_username@eniac.cs.hunter.cuny.edu`
   You can create or navigate through directories there to organize your files (see review of shell commands below), and then upload your files by typing:
   `put filename`
   To upload multiple files at the same time you can use:
   `mput filename1 filename2 …`

2. Once you have uploaded all your files type **`exit`**, that will bring you back to your local machine.

3. To compile and run your code on the lab machine you must first ssh into the server by typing:
   `ssh your_username@eniac.cs.hunter.cuny.edu`
   Username are case sensitive.

4. **type your password** (note that when entering a password no characters will appear on the screen). Passwords are case sensitive.

5. Now you are at a gateway machine that is called eniac. Do not do any processing on eniac. Just ssh through eniac to one of the machines in the lab by typing:
   `ssh your_username@cslabX.cs.hunter.cuny.edu`, where X is a number 1 through 29. You can pick any machine. If the machine is down you can try another machine. For instance, to login to the 2nd machine type:
   `ssh your_username@cslab2.cs.hunter.cuny.edu`

6. All cslabX machines and eniac see the same directories for your account. That means that you see the same files on all machines.

7. Now that you are logged into a Linux machine in the lab you can remotely compile and run your program with g++ (see the next section).

To learn more about logging in remotely, using Linux, following the lab rules, and dealing with possible issues, visit http://compsci.hunter.cuny.edu/~csdir/

Compiling your code with g++

Separate compilation: We are now working with multiple source files that must be compiled into a single executable. Assume your programming project consists of the following files: ClassA.hpp, ClassA.cpp, ClassB.hpp, ClassB.cpp, program1.cpp, main.cpp

You compile only the .cpp files.

To compile your program with g++, at a terminal window type:

```
g++ -o myprogram –std=c++17 ClassA.cpp ClassB.cpp program1.cpp main.cpp
```

This will produce an executable file named `myprogram`. To run the compiled program type in terminal:

```
./myprogram
```

Alternatively, if you compile the program without giving the output file name (leaving out the command option -o *myprogram*), the executable file will be called a.out, which you can execute the same way:

```
./a.out
```

Makefile: we will distribute a `Makefile` with the starter code for our projects. You must understand Makefiles in order to modify them for testing your code before submission according to your needs. If you have never used Makefiles before, here some resources:

- A Short Introduction to Makefile (by Zhiliang Xu)
- Understanding Make (by Alex Allain)

A very quick review of some shell commands:

You need to know just a few commands to work comfortably in a Unix terminal:
`ls, cd, pwd, mkdir, cp, mv, rm.`

A brief summary:

`pwd`	print the current working directory
`ls`	list files in the current directory
`ls path/to/a/directory`	list files in the directory
`cd path/to/a/directory`	change directory

These are some useful directory shortcuts:

`.`	the current directory
`..`	the parent directory of the current
`~`	the home directory

For example:

`cd ..`	go to the parent directory (one level up)

`mkdir newdirectoryname`	create new directory
`cp file1 file2`	copy `file1` and call the copy `file2`
`mv file1 file2`	rename (move) `file1` to `file2`
`rmdir directoryname`	remove empty directory
`rm file`	remove `file`
`chmod <options> file`	change file permissions (read `+r`, write `+w`, execute `+x`)
`man command`	documentation about the `command`

Here are some useful references:

Unix tutorial: http://www.ee.surrey.ac.uk/Teaching/Unix/unix1.html

Become a Command Line Ninja: https://lifehacker.com/5743814/become-a-command-line-ninja-with-these-time-saving-shortcuts