CSCI 235

Software Design & Analysis II

Hunter College - Summer 2019

Programming Rules:

It is **extremely important** to follow these rules. **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 programming assignment for details specific to individual programming projects.

All programming projects must be submitted on **Gradescope** 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, seek help from the TAs immediately during tutoring sessions. The first programming project is due on February 5th.

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

All programming projects must be submitted by **6pm 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. Before submitting to Gradescope you MUST ensure that your program compiles (with g++) and runs correctly on the Linux machines in the labs at Hunter. That is your baseline, if it runs correctly there it will run correctly on Gradescope, and if it does not, you will have the necessary feedback (compiler error messages, debugger

or program output) to guide you in debugging, which you don't have through Gradescope."But it ran on my machine" is not a valid excuse for a failed submission.

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 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 — NOTE: 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 purpose, 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. Although comments will not always directly affect your project grades, you should still thoroughly comment your code. Should you ask for a regrade, I will not regrade uncommented code. **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. 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 by the due date following this link:**
http://www.geography.hunter.cuny.edu/tbw/CS.Linux.Lab.FAQ/department_of_computer_science.faq.htm

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

1. **ssh <*your_username*>@eniac.cs.hunter.cuny.edu**
2. type your password
3. Now you are at a gateway machine that is called eniac
4. Do not do any processing on eniac. Just ssh through eniac to one of the machines in the lab (see next step)
5. **ssh <*your_username*>@cslab<*X*>.cs.hunter.cuny.edu**, where <X> is the 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 cslab<X> machines and eniac see the same directories for your account. That means that you see the same files in all machines.
7. To test your programs on one of the cslab machines you can use **sftp** in order to transfer your code to eniac.
   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 there
   put *local-path* [*remote-path*]
   Note: [remote-path] is optional if you are already in the remote directory where you would like to copy your files, local-path is just the file name if you are already in the local (on your computer) directory that contains your files.
   Once you have uploaded all your files, type **exit**, and then ssh to eniac (see step 1) and after that **ssh** to any cslab<X> machine (see step 5) to compile and run your code there with g++ (read the next section).

To learn more about logging in remotely, using Linux, following the lab rules, and dealing with possible issues, visit **http://www.geography.hunter.cuny.edu/tbw/CS.Linux.Lab.FAQ/department_of_computer_science.faq.htm**

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.h, ClassA.cpp, ClassB.h, ClassB.cpp, program1.cpp, main.cpp

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

```
g++ –o myprogram 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
```

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