CSE100 Algorithm Design and Analysis
Lab Policies and Guidelines, Fall 2018
Last update: September 24, 2018

In each lab assignment, you will be asked to implement a specific algorithm that you have learned in class (in most cases). *You must follow the input/output formats* that are specified in each lab assignment description. We will provide an example in the description, but for more examples, please see `t*`, `o*` files under `testfiles` directory you can find after unzipping `labXX.zip`. Each `t`x file contains an input, and the corresponding output is in file `o`x.

Submission Before the posted deadline, submit your source code through the assignments page of CatCourses. *You must submit only your cpp file (which includes the main function) and the file name MUST be* ***`(your ucm email id).cpp`***. If your name is John Smith and your email address is `jsmith5@ucmerced.edu` you should submit `jsmith5.cpp`. DO NOT ADD or REMOVE information/characters, e.g. as `jsmith5-lab05.cpp` or `jsmith512302123045.cpp` or `jsmith.cpp`.

You arre allowed to resubmit as many times as you want before the deadline, however, **NO LATE SUBMISSIONS** would be allowed afterwards.

Grading We will compile your code using the `GNU C++` Compiler[1] against `C++ 2011` standard[2]. You can program your code in your favorite `IDE`, but make sure you compile it using `GNU C++` compiler with `C++ 2011` standard and **test your code in the Linux Lab**; Use SSH if necessary. We will give you a testing toolkit.

We will run an **automatic** grader to test the correctness of your code. We will test it against number of examples, *some of which* will be shared with you. Your score will be proportional to the number of the test examples your code passes. If you do not see your score there within a week after your submission, please let your lab session TA know.

Please check your code on the Linux Lab machines before submission.

Academic Integrity While we encourage you to discuss your work with other students, the homeworks, lab assignments and exams must be the result of your own work without collaboration. Cheating causes two problems: you learn less well, and it is unfair to students who put honest effort into their work. See the Academic Dishonesty Statement in the syllabus and the UC Merced Academic Honesty Policy. Importantly: *should copying occur, both the student who copied work from another student and the student who gave material to be copied will both automatically receive a zero for the assignment.* ~~A repeated incident will directly lead to an F.~~ **A single incident will directly lead to an F.**

Specifically regarding the lab assignments:

- We use a plagiarism checker. Do not risk failing the course. The lab assignments are simple and instructive and you will learn a lot if you work through them.
- About using the web to get information: do not try to find a solution in the web for the algorithm we ask you to implement and then try to adapt it to the assignment given; that is *cheating*. But do use the web to check the syntax of `C/C++/STL`, `Unix` commands, etc.

[1] `https://gcc.gnu.org/`
[2] `https://en.wikipedia.org/wiki/C++11`

For both lab assignments and homeworks: you must disclose whatever sources you used to complete your work or to help others complete theirs. For homeworks, write it at the beginning of your submission. For programming assignments, write it as a comment at the beginning of your `.cpp` file. Examples of possible sources:

1. I used the textbook, notes taken during the lectures, companion material from the textbook.
2. I helped student X or received help from X (describe how specifically).
3. I consulted an online resource: website / book / video / etc. Give its URL and explain how you used it.
4. I consulted book X.
5. etc.

How to Test your Code by Yourself You will be provided with the same automatic grading tool we use. The only difference is that we will disclose only some examples while the entire test suit has many more. By checking your code with the grader before you submit, you will be able to know if your output format is correct.

Disclaimer. We have tested this script in a few different systems, but we can ONLY guarantee that it works in the CSE100 lab computers, so ensure you test your code there before you submit it.

You need to use Linux Terminal, Mac Terminal (for Mac users), or Cygwin (for Windows users) with properly installed `GCC`. The recomended way is to use UCM Linux machine which is also available online using SSH. To test your code, unzip `labXX.zip` to your working directory and go inside the `labXX` folder. You will see several files. File named `Grader.sh` is the file you will use to test your code. After you finish writing your solution, compile it in terminal using:

```
$ g++ -std=c++11 -o a.exe source_file.cpp
```

Then you can run grader using:

```
$ ./Grader.sh
```

If you implementation is correct, you will see the following output:

```
Test 1: correct.
Test 2: correct.
Test 3: correct.
Test 4: correct.
Test 5: correct.
Total correct: 5/5
```

If you want to test your code for just one test file, you can try:

```
$ ./a.exe < testfiles/t1
```

Troubleshooting and FAQ.

Q: `./Grader.sh: Permission denied.`
A: You need to change file permissons to be executable:

```
$ chmod +x ./Grader.sh
```

Q: `./Grader not found.`

A: We have updated the grader file, now it should have `.sh` ending, therefore use `./Grader.sh`

Q: I have changed my solution and uploaded new one to catcources, and system added a number to the file name. Is it all right?

A: Don't worry, just make sure you are entring right `UCM user name`, i.e. `jsmith5.cpp`, everything else will be handled automatically.

Q: My solution is exactly right, but grader says it is incorrect. What should I do?

A: Remove or add `endl`, make sure there is no additional outputs from your solution.

Q: How can I make sure that my code will get 100% on the automatic grader?

A: Step by step:

1. Make sure your code has proper `C++` syntax without undefined behavior.
2. Make sure your code compiles on our Linux Lab machines. Use SSH if necessary.
3. Make sure your code implements asked algorithm correctly, respecting all edge cases.
4. Make sure your code's output is in a correct format.
5. Make sure your code passes all test cases supplied with Lab materials.
6. Make sure you are submitting with the right name. See **Submission** for details.

That will ensure you get all points from shared test set. We also maintain secret test set that has more tests. Your final grade is sum of public and secret tests passed. If you implemented correct algorithm that respects all edge cases and has expected run time then your solution will get 100%.