ECE 244 Programming Fundamentals Fall 2022

Laboratory Information Handout

This handout provides general information on the laboratory component of the course. Please read it and keep it for future reference during the term.

Assignments

There are five laboratory assignments in this course. The assignments consist of programming exercises using the C++ programming language. Each assignment is to be done *individually* (i.e., not in groups). A handout for each assignment will only be posted on the web page of the course. The assignments and their due dates are listed below. Assignments are always due at ***5:00pm ET*** (i.e., local Toronto time) on the due date.

Assignment	Topic	Due Date
1	The C++/ECF development environment	September 23
2	C++ classes	October 7
3	C++ I/O and arrays	October 28
4	Classes and linked lists	November 25
5	Inheritance	December 7

The assignments have been designed so that there is ample time to completely finish an assignment in the weeks it is allotted. It is your responsibility to correctly identify and meet the deadline for each assignment. *Assignments will not be accepted after their due dates, without prior arrangement with the course coordinator. In general no deadline extensions will be granted except in severe circumstances.*

Labs and Lab Access

All lab assignments are done on the Engineering Computing Facility (ECF) machines (SF1012 and SF1013). There are regularly scheduled lab sessions for ECE244 students, and you are strongly encouraged to attend your scheduled lab session; TAs will be available to answer any questions and offer help during these periods. Lab access and use are possible any time the labs are not occupied or used by another course, but no TAs will be available.

You may also remotely access ECF machines from your home computer through your Internet Service Provider (ISP). ECF access is possible by connecting to an ECF machine using a secure shell (`ssh`) or a `VNC` client. The handout titled "Remote Access to ECF" describes the process in details. You should read this handout ***before*** the first lab and make sure you can access ECF.

It is also possible to work on your assignments on your home computer (i.e., without remotely accessing ECF) using the `ECE244VM` virtual machine. The handout titled "The ECE244VM Virtual Machine" details how to do so. If you decide to take this route (or use your local machine, e.g., with Visual Studio), then please keep in mind that ***you must get your code to work correctly on ECF, no matter where you develop it. A program that does not work correctly on ECF, even if it works correctly on your home machine, will be marked as incorrect! Plan ahead, and give yourself the time to test what you developed at home on ECF before the deadline.***

Getting Help

In general, the TAs will be available in-person only during scheduled lab sessions. This makes it difficult to get help at an arbitrary time when some problem arises. To alleviate this difficulty, we created a discussion forum for each lab assignment on the popular website ***Piazza***. You may follow this link to enrol in the ECE244 course on Piazza using your email address:

https://piazza.com/utoronto.ca/fall2022/ece244

The TAs and the instructor will regularly check this discussion forum to answer questions. The answers become available to all students. Thus, it pays to check the forums before posting a question to make sure that the question has not been posted and replied to earlier.

You are encouraged to use the forums, and to engage in discussions about the lab assignments with fellow students**.** ***Do not post code on the forums***. Doing so will be treated as an academic offense (see below). If you must include code with your questions, direct your question on Piazza to only the instructors.

Testing Your Code

The handout for each programming assignment will specify the required functionality of the assignment. The specified required functionality is referred to as the ***specification***. It is up to you to make sure that your solution delivers this functionality and adheres exactly to the specification. This is best done using a set of input test cases and checking to make sure that the output is "correct". It is your responsibility to come up with test cases that will cover the specification. Some test cases will be given in the assignment handout, but the set of test cases used to mark your assignment will not be made available to you ahead of time.

There is a utility program called **exercise** to help you test your code. The program will test your code against a few, but not most, of the test cases that will be used to mark your code (more on this below). For more information on the exercise program and how to use it, please refer to the Lab 1 Handout.

It is important to note that you must read the specification provided with the assignment carefully and precisely. Not doing so will likely cause extreme frustration. For example, if part of the assignment requires your program to output an 'a' followed by a blank and a 'b', and your program outputs two blanks between the 'a' and the 'b' (which may be difficult to see on a screen), then

you will likely get zero marks for that part of the assignment, even though your program is "almost" correct.

Submitting Your Work

Before the assignment deadline, a program satisfying the assignment specifications should be demonstrated to work, and then submitted for marking using a command called **submit**. This command has the following syntax:

~ece244i/public/submit <assign-number>

The command looks in the ***current directory*** for all the files required for submission for assignment number *assign-number*. All the files required by the assignment must be in this directory, or **submit** will complain. Submit will also compile your program and run some tests on it; pay careful attention to the messages from **submit**, as any error messages indicate a problem with your code or the submission, and will result in you losing some (or often all) of your marks. Further details on the **submit** command are available in the *ECF Unix Environment* document distributed with Lab 1 Handout.

You may submit your code multiple times. However, each time code is re-submitted it replaces previously submitted code. Thus, it pays to be careful.

Independent Work

Each lab assignment is to be done by each student *individually*. Students are encouraged to discuss with one another issues and problems that arise in the course of solving the laboratory assignments. It is often the case that the answer to a simple question asked of a fellow student can remove a great burden of mystification. The comparison of various approaches to a problem will often reveal difficulties or a common ground that make such discussions invaluable.

However, **work submitted for credit by a student must be the student's own work**. It is one thing to discuss and compare, but quite another to rely on some other student's work to obtain credit for an assignment. It is also an offense to knowingly allow a copy of your work to be submitted by another person for credit. It is also an offense not to put in place protections to prevent your code from being copied without your knowledge!

A reasonable rule of thumb to follow during a discussion is that nobody leaves the discussion with written notes of what was said. It is unlikely that two people who have discussed various approaches to a problem will write highly similar programs unless one or both have a written record of what was said.

All programs submitted for credit in this course will be compared pair wise to attempt to identify cases of collusion, copying, and similar offenses. A utility capable of detecting similar programs, even if considerable effort has been taken to conceal their similarity, is used to compare programs.

Any work submitted for credit that is not the work of the person submitting it will be treated as an offense under the Code of Academic Discipline of the University. Similarly, aiding anyone in the submission of work that is not the work of the submitter will also be treated as an offense under the Code of Academic Discipline of the University. The Code of Academic Discipline will be rigidly enforced in this course. Penalties can range from grade penalties in the course to suspension from the University. The Dean, as advised by the instructor and the Departmental Chair, determines the penalty for each case.

Marking

Your submitted code will be marked for functionality. An automated testing script (called **autotest**) will be used to determine if your program behaves correctly against a set of input test cases. This is done by comparing the output of your program with the "correct" output of the program for each input. This is why it is imperative that you adhere to the output format specified in each assignment handout exactly.