ECE 244 **Programming Fundamentals**

Laboratory Information

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 several laboratory assignments in this course. The assignments consist of programming exercises using the C++ programming language. A handout for each assignment will be posted on the web page of the course. (***No copies will be provided in class.***) The assignments and their due dates are listed below. Assignments are always due at ***5 pm*** on the due date.

Assignment	Topic	Due Date
1	The C++ development environment	September 21
2	A generic command interpreter	October 5
3	Basic C++ classes	October 19
4	Midterm	October 30
	C++ classes: access control	November 9
5	C++ classes: graphs	November 23
6	Inheritance	December 5

The assignments have been designed so that there is ample time to completely finish the assignment. 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 labs will be in SF 1013. There are regularly scheduled lab sessions for ECE 244 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 use your home computer to work on assignments. You may do so by remotely accessing ECF from home through your Internet Service Provider (ISP). To do so, you must connect to ECF using a secure sell (ssh). For more information on using ssh and on using ECF remotely, please consult http://www.ecf.toronto.edu/ecf/ssh.html. You may also use your home computer by downloading the programs you need (e.g., compiler, debugger, etc.) directly to your computer. Information on how to do is provided in the first laboratory assignment handouts.

If you decide to work on your assignments on your home computer, then ***it is important to note 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 TA's will be on duty only during scheduled lab sessions. This makes it difficult to get help at an arbitrary time when some problem arises. To alleviate this difficulty, there is a discussion board section of course's Web site. The TA's as well as the instructors will regularly check this board to answer questions. The answers become available to all students. Thus, it pays to check the board before posting a question to make sure that the question has not been posted and replied to earlier.

You are encouraged to use the discussion board, and to engage in discussions about the lab assignments with fellow students**. *Do not post code on the discussion board***. Doing so will be treated as an academic offense (see below). Also, if the message is to be directed to a specific TA or instructor, the use of e-mail is more sensible

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, some will be posted on the course's discussion board, but the set of test cases used to mark your assignment will not be made available to you ahead of time.

There exists 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 Labs section on the course Web site.

It is important to note that you 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 and '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 **submitece244f**. This command has the following syntax:

Submitece244f assign-no file*1* file*2* … file*n*

The command submits your files: file*1*, file*2*, … file*n*, in the current directory, for assignment number assign-no. Submit only source files and the Makefile so that typing "make" will correctly generate your executable. For example, to submit your solutions to assignment 3, which consist of "main.cc", "tree.cc", "treenode.cc" and "Makefile", use the command as follows:

Submitece244f 3 main.cc tree.cc treenode.cc Makefile

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.

There exists a utility program called **chksubmit**, which will aid you in ensuring that you have submitted all the files necessary to mark your assignment. Information on how to use the chksubmit program will be provided on the course Web site.

All source code (including the Makefile) must be compiled and submitted on ECF computers with a "p*n*.ecf.utoronto.edu" name, for example "p85.ecf.utoronto.ca", or "p112.ecf.utoronto.ca". These are the Linux machines located in SF 1012 and SF 1013. You can tell which machine you are using from the command prompt. For example, if you are using p110.ecf.utoronto.ca, the command prompt will read p110.ecf). Do not compile your work on skule skule.ecf or remote.ecf. Doing so will result in problems with compilation and/or submission.

Independent Work

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 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 sophisticated program that is capable of detecting similar programs even if considerable effort has been taken to conceal their similarity does the comparison.

Marking

Your submitted code will be marked for both functionality and style. 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.

The TAs will also inspect your submitted source code. Comments on programming style and structure will be returned on a marked up copy of the source program. Good programming style is expected and required. Guidelines on good programming style will be provided on the course Web site. A style mark will be assigned, which is worth 20% of the assignment's total mark. Exceptionally well-structured programs, with outstanding style and documentation, may be judged as good enough by the TA to warrant an additional 10% style mark.

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.