CS 240

Data Structures and Algorithms

Spring 2014

1 Lab 00

In this course you will be learning how to program in C++. Programming experience is a pre-requisite for this course,

2 Setup (Review From Class)

The labs at Binghamton University, in `LN G103` are Ubuntu-based. There is also a Binghamton University Linux server that we can utilize in order to prevent you from having to physically be in the `LN G103` lab in order to do work.

To connect to this server from your home machine you will need to use SSH. For Windows users, you can get the SSH client from the University's FTP server, ssl.binghamton.edu, where you log in with your PODS username and password to the PODS domain and then click on "BU FTP Site" -> "windows" -> "ssh2" -> "SSHSecureShellClient-3.2.9.exe". Alternatively you can use PuTTY, which you can get from the PuTTY Download Page, http://www.chiark.greenend.org.uk/ sgtatham/putty/download.html, or from the BU FTP Site mentioned above. For Mac and Linux users, there is an SSH client built in that can be accessed from a terminal session by typing `ssh username@hostname`. Linux users may have to install this utility manually.

The hostname of the server is `harvey.cc.binghamton.edu`. Your account credentials are the same username and password that you use to log into the `PODS` machines and any other Bingsuns account you may have. If you are unable to connect to Harvey, you may have to chain-SSH through `bingsuns.cc.binghamton.edu` in order to obtain an on-campus IP address.

You may find during the semester that you exceed your quota. This can happen if you are storing a lot of files on the Bingsuns server. If this is the case, we will not be able to help you, so it is your responsibility to delete files that are not needed so that you remain below your quota. You can check how close you are to the quota by executing the `quota` command from a shell that is connected to `bingsuns.cc.binghamton.edu`. Exceeding the quota will make it so that you are unable to save your changes to files, compile your code, or sometimes it may prevent you from logging into the machines during the lab session. Failure to maintain your quota could waste your time during the lab – resulting in a lower grade than you might expect.

You can use the manual entry for most commands in Linux in order to figure out what they do. Some simple Linux commands that you should already be familiar with are:

- ls
- cd
- rm
- cp
- mv
- tar
- pwd
- g++
- make
- gdb
- scp
- cat

You will also need to either use FTP utility (either one of the SSH utilities listed above, or something like FileZilla) in order to transfer files between your local machine and the remote server. If you do not want to learn to use

an FTP utility then you can learn to utilize a command-line editor such as nano, vim, or emacs. Each of these editors has some learning curve to them, but often the more steep the learning curve the more powerful the editor. During the lab sessions you may not have to use this editor (although it will still be available) and you can instead use a standard GUI-based file editor like gedit.

3 Header Files

Once you begin the lab assignment, you will be required to implement some specific functionality. To make the programs modular, you may be given what is called a header file. Header files contain the declarations of functions, constants, variables, etc., and they usually have the name `filename.h`.

Header files often have what are referred to as Include Guards. These help to prevent multiple declarations and define scope. Header files are often laid out in such a way that the class is defined, including its functionality. This may look something like this:

```
class myClass{
     private:
          /*private data members and methods declared here*/
     public:
          /*public data members and methods declared here*/
};
```

Just like in Java, you may have to import functionality, whereas Java uses `import` statements, C++ uses `#include` statements. You may have to Google to figure out which file to include.

4 Implementation Files

Header files usually have a one-to-one correspondence between their associated implementation files, usually having the name of either `filename.cpp` or `filename.cc`. And most often the implementation files have include statements for their associated header files. For the `myClass` example, the header file would be named `myClass.h` and the implementation file would contain the line `#include "myClass.h"`. Implementation files are usually compiled separately and then linked together to produce a binary executable.

These files are where you will write your code. C++ has a scoping technique in order to differentiate between a function and a method. In general, if a class has a method, the method name will have to be scoped using the `::` scoping operator. Suppose you have a class `myClass` and it has a method `myMethod` with a void return type and an integer parameter, then your implementation file will look like:

```
void myClass::myMethod(int param){...}
```

If you are unfamiliar with the difference between a function and a method, this is an OOP concept that you should have learned, so you will need do a little self-study to figure out the difference.

More information on this can be found in Appendix C of the Nyhoff book.

5 Makefile

For this lab you will be provided with a makefile. These files will either need to be named `Makefile` or `makefile`. The Linux make utility uses the rules defined in this file in order to determine what to compile and how to compile it. Use the manual entry for make (accessed by executing the command `man make`) in order to understand how this utility works. For more information, you might want to check out the make tutorial found here, http://mrbook.org/tutorials/make/.

6 GNU Debugger

If you are having problems figuring out why your code crashes, you can use gdb. Please feel free to use the tutorials for gdb that are available online, such as this one, http://www.cs.cmu.edu/ gilpin/tutorial/. This requires a special flag when compiling, so you should be sure that this flag is set in your makefile.

Using the GNU Debugger will be useful throughout your career in CS. Using this tool will be required as part of a later lab, should you find that you are overwhelmed with debugging the code for this assignment, please let me know early and I will do what I can to assist you without you having to learn all of the tools presented here.

7 The Questions

For each question, before you look at the associated header file that you are to implement, you should consider what your method signature will look like. This will help you to solidify what each question is asking before you begin answering.

Question 1

Write a method that takes four parameters. The first parameter is an array, the second parameter is an integer value (the size of the array), the third parameter is an integer N, and the fourth parameter is an integer M. Your function should print all values in the array, except for those that have a value equal to N. Instead of N, at each of these locations you should print M. Each value should be printed separated by a space. For example, if the parameters are: [1,2,3,2,1] 5 2 6 then the output will be:

```
1 6 3 6 1
```

Question 2

Write a method that prints out only the even numbers in the range from A to B (exclusive), one per line. Note that this means A and B must be parameters to your method. Also note that by definition, a number N is even if `N = 2*Z`, for some value of Z in the set of integers.

Question 3

Write code to find the smallest common multiple of two integers. The parameters for this method should be two integers. In the event that no such multiple exists, you should return `-1`.

Question 4

Write a method that takes two boolean parameters. This function will print out the evaluation of logical predicates, one per line, labeled in all caps and the evaluation separated by one space. The predicates are as follows: `AND, OR, NAND, NOR, XOR, IMPL, EQU` Note that AND is a logical and; OR is a logical or; NAND is the negation of AND; NOR is the negation of OR; XOR is an exclusive logical or; IMPL is a logical implication; EQU is a logical equivalence.

For example, if the parameters are: true, false then the output will be

```
AND false
OR true
NAND true
NOR false
XOR true
```

```
IMPL false
EQU false
```

Question 5

Write a program that prints the numbers from 1 to 100, one per line. For multiples of five, instead of the number, print the word `Fizz`. Similarly, for multiples of nine, write the word `Buzz`. For numbers that are multiples of both five and nine, write the words `FizzBuzz`.