EECE.4810/EECE.5730: Operating Systems

Spring 2020

Individual Program 1

Due **11:59 PM**, **Friday, 2/7/20**

1. Introduction

This program introduces simple fundamentals of programming on UNIX systems. You will review a few fundamental data structures, practice having your program read command-line arguments, and learn how to write a simple makefile.

This assignment is worth a total of 40 points. The given grading rubric in Section 4 applies to students in both EECE.4810 and EECE.5730.

2. Project Submission and Deliverables

Your submission must meet the following requirements:

- Your solution may be written in C or C++.
- You should submit a .zip file *(no other archive format)* containing the following files:
    - All source/header files you write. You should have at least three such files:
        - A source file containing your `main()` function
        - A header file for any structure/class definition(s) and function prototypes (if you use multiple structures/classes, you'll have multiple source files)
        - A source file for your function definitions (again, multiple structures/ classes means multiple source files)
    - A makefile that can be used to build your project
    - A README file that briefly describes the contents of your submission, as well as directions for compiling and running your code.
- Programs must be submitted via Blackboard.
- You must work individually on this program—this is not a group assignment.

While you may write your code on any machine you like, your program must run on one of the machines in Ball 410, which everyone should be able to access via keycard.

- Your login details for each machine are as follows:
    - Username: 1st initial + 1st 8 chars of last name—all lowercase (e.g., `mgeiger`)
    - Password: 1st/last initial (uppercase) + "@bl" (lowercase B, lowercase L) + *num*
        - *num* is usually (but not always) 416
        - For example, `MG@bl416`
        - **I will email your password to you—DO NOT CHANGE IT**

Ball 410 login details (continued)

- Remote access (*requires VPN access when off-campus—visit vpn.uml.edu*)
  - Via shell: `ssh <username>@anacondaX.uml.edu` (*X* = 1, 2, 3, …)
    - For example: `ssh mgeiger@anaconda1.uml.edu`
  - Via X2Go application—see course home page for details

3. Specification

General overview: Your main program should maintain three different data structures—a queue, a stack, and a linked list. Your implementation of each data structure should not impose a maximum number of values in each.

The program will read a set of integers from a file and copy those values into all three data structures, using appropriate insert methods so that:

- The queue contents exactly match the order of values in the file, with the first value from the file at the front of the final queue.
- The stack contents are stored in the opposite order, since a stack is a last-in, first-out data structure. The top of the final stack will be the last value read from the file.
- The linked list contents should be ordered from lowest to highest value.

Input: Your program takes input from two sources:

- *Command line arguments:* This program takes a single command line argument, the name of the text input file containing the integer list. See the lecture slides/recording from Wednesday, 1/29 for more details on reading command line arguments.

  If your executable name is `prog1`, you might run it with command line:

  ```
  ./prog1 file1.txt
  ```

- *File input:* The input file contains a series of integer values, one per line. You can assume there will be no errors when reading the file contents. One sample file will be posted with the assignment, and the results of reading it are shown below.

Output: After reading the file, your program should print the contents of all three structures. For the stack, start at the top of the stack; for the queue, start at the front of the queue.

Each data structure's contents should be printed on a single line, with one space between values.

For example, given the sample input file `file1.txt` posted with the assignment, your program should print the following lines of output:

```
QUEUE CONTENTS:
48 10 57 30 -5 5 1 31 20 -9
STACK CONTENTS:
-9 20 31 1 5 -5 30 57 10 48
LIST CONTENTS:
-9 -5 1 5 10 20 30 31 48 57
```

Makefiles: On the Linux machines in Ball 410, you should use the C compiler `gcc` or C++ compiler `g++` to compile your code. Repeated compilation is most easily done using the `make` utility, which requires you to write a makefile for your code. Reference material for writing makefiles is easily found online; I found the following website to be a good basic makefile introduction, which is all you should need for this assignment:

http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

Remember, as noted above, a working makefile must be part of your submission, and instructions on its use must be included in your README file.

4. Grading Rubric

Due to the simplicity of this assignment, the grading rubric is relatively simple as well:

- **10 points**: Your program compiles successfully using the makefile you provided.
- **10 points**: Your program implements a proper queue and prints its contents correctly.
- **10 points**: Your program implements a proper stack and prints its contents correctly.
- **10 points**: Your program implements a proper linked list and prints its contents correctly.