EECE.4811/EECE.5811: Operating Systems

Spring 2022

Individual Program 1

Due **<u>5:00 PM</u>, Friday, 2/4/21**

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 program build system using either `make` or `cmake`.

This assignment is worth a total of 40 points. The given grading rubric in Section 4 applies to students in both EECE.4811 and EECE.5811.

2. Project Submission and Deliverables

Your submission must meet the following requirements:

- Your solution may be written in C or C++.
- You must submit an archive file *(<u>.zip, .tar, or .tgz format only</u>)* containing:
  - All source/header files you write. You may 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 (multiple structures/classes have multiple files)
    - A source file for your function definitions
  - Appropriate build files for your project—either a makefile to be used with `make`, or appropriate files to be used with `cmake` *(see details on page 3)*
  - 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.
- <u>You must work individually on this program—this is not a group assignment.</u>

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 remotely.

- Your login details for each machine are as follows:
  - Username: 1st initial + 1st 7 chars of last name—<u>all lowercase</u> (e.g., `mgeiger`)
  - Password: 1st 4 letters of last name + *num*
    - *num* is the first 4 digits of your UML ID number <u>after any leading 0s</u>
    - `For example, geig1234`

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* = 2, 3, …)
        - For example: `ssh mgeiger@anaconda2.uml.edu`
    - Via X2Go application—see course home page for details

3. Specification

General overview: Your main program must maintain three different data structures—a queue, a stack, and a linked list. Your implementation of each structure should not limit the number of values it stores. You may use C++ STL classes if you want.

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 stack is 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. We discussed command line arguments in detail in class on Wednesday, 1/26.

  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. Start at the top of the stack and the front of the queue when displaying output from those structures.

Print each data structure's contents on a single line, with one space between values.

For example, given the sample input file `file1.txt`, your program should print:

```
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
```

Building your code: 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 one of two utilities:

- `make`, 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/

- `cmake`, which requires you to specify a file similar to a makefile (`CMakeLists.txt`), then use that to set up a build directory and build your program. The CMake home page contains a detailed tutorial at the link below. You can get by using just the first couple of steps—if you specify all of your source files to the `add_executable` command, all of those files will be compiled to generate your executable:

  https://cmake.org/cmake/help/latest/guide/tutorial/index.html

Remember, as noted above, a working build system for `make` or `cmake` 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 using the build system you provided.
- **10 points**: Your program implements a queue and prints its contents correctly.
- **10 points**: Your program implements a stack and prints its contents correctly.
- **10 points**: Your program implements a linked list and prints its contents correctly.