Project 1 - A Stack in C

CS120 Spring 2020

**Due Date**  
Midnight, Monday, April 13th

*All programs will be tested on the machines in the G7 lab. If your code does not run on the system in this lab, it is considered non-functioning EVEN IF IT RUNS ON YOUR PERSONAL COMPUTER. Always check that your code runs on the lab machines before submitting.*

# Driver Code and Test Files

* project1.c
* mylib.h
* README.md

# Grading Rubric

**TOTAL: 40 points**

* Part A: *36 points*
  + Each Test Passed: 1-12 (3 points)
* Part B: *10 points*
  + No Memory Leaks or Errors reported by Valgrind
* Part C: *4 points*
  + All code is in mylib.h file (2 points)
  + Follows requested project structure and submission format (1 point)
  + README is complete (1 point)

# 

# Guidelines and Policies

## Getting Help

Please follow the debugging guidelines outlined [here](https://docs.google.com/document/d/1gmuVuHQn8rK9p1Clt-SospDACtOAOiy9UPhu-8rPhGg/edit?usp=sharing). We will try to answer questions and provide help within 24 hours of your request. If you do not receive a response in 24 hours, please send the request again.

Although we will answer questions, provide clarification, and give general help where possible up until the deadline, we will not help you debug specific code within 24 hours of the deadline. We will not provide any help after the deadline.

## Guidelines

This is an individual assignment. You must do the vast majority of the work on your own. It is permissible to consult with classmates to ask general questions about the assignment, to help discover and fix specific bugs, and to talk about high level approaches in general terms. It is not permissible to give or receive answers or solution details from fellow students.

You may research online for additional resources; however, you may not use code that was written specifically *to* solve the problem you have been given, and you may not have anyone else help you write the code or solve the problem. You may use code snippets found online, providing that they are appropriately and clearly cited, within your submitted code.

*By submitting this assignment, you agree that you have followed the above guidelines regarding collaboration and research.*

# Part A - Creating A Stack

**NOTE:** Each test in the driver code (project1.c) is independent. You can turn on and off tests for debugging purposes by changing the TEST value to 0 in the driver code. For example, to turn off TEST2, change:

|  |
| --- |
| #define TEST2 1 |

to

|  |
| --- |
| #define TEST2 0 |

Make sure **ALL** tests are on when you submit.

## Stacks

In this program you will create a simple program to understand the implementation of Stack in C. A Stack is what we refer to as a LIFO (LAST IN / FIRST OUT) data structure. We’ve already discussed the concept of a Stack, but here are some additional resources to help you understand the basic idea:

* <http://www.c4learn.com/data-structure/stack-concept-overview-definition/>
* <https://medium.com/swlh/stacks-and-queues-simplified-ef0f838fc534>
* <https://www.cs.usfca.edu/~galles/visualization/StackArray.html>

The implementation of a stack requires every new element be inserted as the ‘top‘ element. You also must ensure that you do not go over the maximum size of the Stack.

You will write all your code for this project in the mylib.h file, which is included into the project1.c file.

I have provided you with a struct Stack that you will use to implement the stack functionality.

|  |
| --- |
| struct Stack{  int \* stk;  int top;  int max\_size; }; |

That means every newly inserted element is pointed by top (by index) and the array, stk, must be of max\_size.

You will need to create two object management functions:

* struct stack \* newStack(int size)
  + Creates a new struct Stack object with a max size of size on the heap
  + returns a pointer to the newly created struct Stack
* void deleteStack(struct stack \* s)
  + Takes a pointer to a struct Stack object as a parameter
  + Deallocates the struct Stack object memory on the heap

*Once you are comfortable with the functionality of your stack, you must implement the following functions*:

### pop - Removing an element from the stack

|  |
| --- |
| int pop (struct stack \* s) |

Whenever we want to remove an element from the stack, simply move ‘top‘ to point to the previous item in the array.

* Step 1: Check if Stack is empty
  + If top is -1, that means the stack is empty, return -1
* Step 2: Store the current value at top in a temp variable
* Step 3: Decrease top by 1
* Step 3: return the temporary variable

### push – Inserting an element into the Stack

|  |
| --- |
| int push(struct stack \* s, int value) |

We can use the following steps to insert a new node into the stack

* Step 1: Check if Stack is full
  + If top is equal to max\_size-1, that means the stack is full, return -1
* Step 2: Store value at top in stk
* Step 3: Increase top by 1
* Step 3: return 0 upon success

### display - display your stack

|  |
| --- |
| void display(struct stack \* s) |

Print out the following information about your stack:

|  |
| --- |
| top max\_size |

For each item in stk: <index> : <values>

### stringReverse - use your stack to reverse a string

|  |
| --- |
| void stringReverse(char \* str, struct stack \* s); |

One common use of a stack is to easily reverse a string. Write a function that takes a struct Stack and char array as parameters, and uses the stack to reverse the original string (*you must use the stack, you may not use a library function to reverse the string*).

# Part B: Memory

For Part B, you need to run your code with valgrind to ensure there are no memory leaks or memory errors.

# Part C

## Submission

Required code organization:

* project1.c
* mylib.h
* README.md

Below is just a reminder of the commands you should use to submit your code. If you cannot remember the exact process, please review Lab 0.

*These commands all presume that your present working directory is within the directory tracked by git.*

You will need to do the following when your submission is ready for grading.

|  |
| --- |
| git commit -a -m "final commit" git push |

To complete your submission, you must copy and paste the commit hash into MyCourses. Go to MyCourses, select CS110, and then assignments. Select this lab, and where it says text submission, paste your commit hash. **DO NOT PASTE ANYTHING OTHER THAN YOUR COMMIT HASH.**    
**Incorrect**: “commit hash: 690fa67ed8”.   
**Correct**: 690fa67ed8

You can get your latest commit hash with the following command:

|  |
| --- |
| git rev-parse HEAD |

Remember, you **MUST** make a submission on MyCourses before the deadline to be considered on time.