LAB #9 – A Stack: Array vs. Linked List

Each lab will begin with a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. You can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstances, contact your lab TAs and Jennifer Parham-Mocello.

Pair Programming
In this lab, you can choose a partner for pair programming. You must be checked off together. You only need one computer for pair programming. One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, your TA will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: Pair Programming Student Handout

(3 pt) Debugging Programs
Download the following program, stackdebug.c. This is a program that implements a stack, which is a linear structure with the property that the last item in is the first item out, using a dynamic array. This program implements the addition and subtraction in a reverse polish notation calculator, like the one found on our system.

% dc
3
4
-
p
-1
q

You can read a wiki page on RPN: https://en.wikipedia.org/wiki/Reverse_Polish_notation
In order to see the answer/result on the stack, you have to type ‘p’, and ‘q’ will quit the calculator. There has to be 2 operands on the stack in order to enter an operator.

Your job is to find all the C syntax and logic bugs in this program! Remember, you need to compile with gcc, not g++!!! Have fun!!!

(7 pts) Implement a Stack using a linked list in C
In this lab, you are going to implement a stack data structure using a linked list in C. A stack is a LIFO (last in, first out) structure, where you only push and pop items from the top. For example, a deck of cards, a stack of plates, etc. We will implement the stack using a linked list and a struct holding the linked list, which stores the location of the top and bottom of the stack.

```c
struct node {
    int val;  //integer node value
    struct node *next;  //pointer to next node
};
```
struct stack {
    struct node *head;  //pointer to first node in stack
    struct node *tail;  //pointer to top of stack
};

Begin by implementing the following stack operations:
void init(struct stack *);  //initialize stack members
void push(struct stack *, int);  //grow contents to store int
int pop(struct stack *);  //shrink contents and return top int

When implementing these stack operations, don’t forget about the tail member!

C Hints:
- Include <stdio.h> for printf/scanf
- Include <stdlib.h> for malloc/free and NULL
- Use the -> to access members from a struct pointer
- Use Wednesday's/Monday's notes on the calendar page

You must compile with gcc!!!
gcc stack.c -o stack

Convince yourself and the TAs that the init(), push(), and pop() stack implementations are working using a linked list!!!

A few Questions: Write answers on a piece of paper

- How does the complexity of push() differ between the dynamic array and this linked list implementation?
- What would be the pros and cons of using a static array with a max size as a stack?
- How would the complexity of push() change if you didn’t have a tail?
- How could you make the complexity of pop() better? Change the algorithm or data structure?

Extended Learning/Implement the rest of the stack:
int peek(struct stack);  //Returns the top element in the stack
void destroy(struct stack *);  //destroy all elements in stack
bool empty(stack);  //return true if the stack is empty
int size(stack);  //return the number of elements in stack

Remember, you and your partner will not receive lab credit if you do not get checked off before leaving each lab. Once you have a zero on a lab, then it cannot be changed because we have no way of knowing if you were there or not!!!