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

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.

Reminder: All of our labs involve paired programming. You do not have to keep the same partner for each lab, but you MUST work with someone in each lab!!! First, find a partner for this lab. It can be the same partner from the previous lab or a different partner.

(1 pt) Canvas: Consent Form

Go to your COURSE (not lab) section in Canvas. Go to the Quizzes link on the left, and answer the two questions in the ungraded Consent Form. Make sure you go to your COURSE (not lab!!!) section.

(7 pts) Implement a Stack 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:

```c
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!!!

(2 pts) 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:

```c
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!!!