CS162 Lab 6

Goals

- Build linked structures using **pointers**

In this lab, you will create simple structures using **dynamic memory**. You will implement a Stack class and a Queue class. Each will have a pointer to the appropriate struct and the designated functions.

1. **Stack-like structure**. A stack is a **first in last out** (FILO) structure. So the top of the stack will be the last item entered. You are not required to implement any other features of the formal Stack data structure. Be careful when you research online. You could make your program more difficult.

You will create a **single linked Stacknode**. You will need to add an element and to remove an element. You should have functions:

```
void push(Type_of_Data Value)
Type_of_Data pop()
```

`Type_of_Data` is whatever data type you want to use (int, float, char, string, or even a class). You are only required to use one data type! So using int, `void push(int Value)` will take an integer value, create a node, set your value to the data part of the node, and adjust pointers as necessary. The `pop()` function will return the data in the top node and also remove the top node and adjust pointers. This is all that is required.

2. **Queue-like structure**. A queue is a first in first out (FIFO) structure. You will need a pointer to the front and to the back of the queue. You are not required to implement any other features of the formal Queue data structure.

You will create a **doubly linked Queuenode**. You will need a front pointer and a back (or rear) pointer. You will only need to add an element at the back node and to take off an element from the front node. You should have functions:

```
void push(Type_of_Data Value)
Type_of_Data pop()
```

`Type_of_Data` is whatever data type you want to use. You are only required to use one data type! So using int, `push()` will take an integer value, create a node, set your value to the data part of the node, and adjust the back pointer to point to it. The `pop()` function will return the data in the front node. It will also remove the node and adjust the front pointer.

Your program should firstly use the `push` function to build the stack and queue. You will also need an interface to allow you to choose the operation of push/pop, and enter the value of node to push. You need to consider the situation when you have **no node** in the stack/queue (they are empty) and you try to pop. Your pop function should be able to handle that.

A recommend interface flow would be: allow user to choose stack or queue to operate, then ask user to choose from push/pop operation; if user choose push, ask for the input value (you need to tell user what kind of data type you use for your stack/queue) and do data validation; if user choose pop, do the stack/queue empty validation and pop. If you do not use this flow, then you must have a reasonable method to build your stack/queue and do manipulation on that.

De-allocate memory as appropriate. Remember that **you cannot use or copy code from any source**. Your work must be your own. You will use your code as part of assignment 4.

For this lab, you need to write a **readme** file to briefly describe how to build and operate the stack/queue, and TA will follow your instructions to run your program. Turn in all your program files, makefile and readme file in a **zip** file on **TEACH**.

Grading

- Programming style: 1 point
- Simple first in last out structure: 3 points
  - Correctly create the single linked Stacknode
  - Correctly implement `push()`
  - Correctly implement `pop()`
- Simple first in first out structure: 3 points
  - Correctly create the double linked Queuenode
  - Correctly implement `push()`
  - Correctly implement `pop()`
- Proper use of dynamic memory: 1 point
- Proper create an interface for manipulate the stack and queue: 2 points