Programming Assignment #2
CS 163 Data Structures

Submit your assignment to the D2L Dropbox (sign on via [d2l.pdx.edu](d2l.pdx.edu))

Goal: The purpose of the second programming assignment is to experience stacks, queues, and new data structures. The data structures for this assignment are a linear linked list of arrays and a circular linked list.

Problem Statement:
I have been traveling frequently between Corvallis and Portland. The drive is pretty easy except when there are problems on I5. When there is a backup, a crash, or bad weather, it is nice to know what alternative routes there are, how long they will take me and where to turn (such as knowing where to catch the Dundee bypass off of highway 99 is essential because the sign's are minimal). Maybe you frequently travel somewhere – where there are multiple ways of getting there. Do you ever wonder which route is best?

For program #2, we will use a both a Stack ADT and a Queue ADT to assist with managing our routes. A route from one location to the next will be stored using a Queue ADT. The application will use at least two Queue instances for two different possible routes. The Stack will be used for the return trip. So, each time we travel through a leg of a route, that information will be pushed on the stack giving the information in a way we can access to return (remember a stack reverses the order of our data making this a great abstraction for this problem).

Programming: There are two ADTs in this program (Queue and Stack ADT). This means each needs to be implemented as a separate class. You can put the class interfaces in either one .h file or two .h files.

1. The information kept about each leg of a trip
   a. Name of the street or leg of the trip (e.g., Highway 34)
   b. Length (e.g., 5 miles)
   c. Current Traffic (e.g., Light)
   d. Notes about the road (e.g., Often foggy at night, avoid if possible)
   e. Add one other item of your choice

- Please keep in mind that because we are creating ADTs, the **<u>user</u>** of the program must not be aware that stacks and queues are being used. However, the client program <u>knows</u> that they are using these ADTs.
- You should support complete implementations of the Stack and Queue ADT. Make sure to thoroughly test each of the stack and queue functions from the client program!

Programming – Data Structures:

- The stack should be implemented using a linear linked list of arrays (lab 3), where each element in the array is a leg or segment in a longer journey. The array must be dynamically allocated and each array must be of the same size. I recommend that the arrays be relatively small (e.g., 5) so that you can properly test the code. Stack ADT functions should include **push, pop, peek, and display.**
- The queue should be implemented using a circular linked list (lab 3 and 4), where the rear pointer points to the last segment we will take, and rear->next points to the first. You must implement **enqueue, dequeue, peek, and display.**
- Remember the idea of your client program is to test out your ADT functionality so that a real application can be developed off site by another team of Software Engineers based on the ADT public functions. You want to make sure that your stack and queue ADTs perform well!

<u>**Things you should know...as part of your program:**</u>

1) **Do not use statically allocated arrays in your classes or structures used by the ADT.**
2) All data members in a class must be private
3) Never perform input operations from your class in CS163
4) Global variables are not allowed in CS163
5) **Do not use the String class! (use arrays of characters instead!) You may use the cstring library.**
6) Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary prototypes. The .cpp files should contain function definitions. **Never "#include" .cpp files!**