Lab 3 Linked Structures

This is **NOT** a linked list!

You will create a queue abstract data type. You will use a linked linear structure to store the values and provide the necessary functions to interact with that linked structure. Data is a character!! You will also add a function to test if the structure is empty.

You will use pointer variable(s) to the element/node that is at the beginning of the structure, and the end of the structure. You will not use a pointer to iterate or go to other individual elements of the structure.

The queue is a first in first out structure. You add to the back and can only look at or take off the front. **Sit down with pencil and paper and study the required pointer manipulations.** You only have the link(s) in each node, a pointer to the front and a pointer to the back. Use a singly linked structure. Have your pointer algorithms written out before you start writing your code! You will implement these functions, with appropriate parameters and return types:

```plaintext
addBack()    // puts on item at the end of the structure
getFront()   // returns the value at the front of the structure
removeFront() // removes the first item in the structure
isEmpty():
```

When you write `isEmpty()` it should not be counting the nodes. How do you do it?

**TESTING**

You must also write a driver program that uses your queue to demonstrate it works correctly. You should prompt the user to enter a series of characters, with some way to indicate they have finished entering values. Just enter the each value into your queue. Then you print out the values as you remove them from your structure.

Your code should also generate an error if a user attempts to remove more items than were entered into the structure. Add to your driver program a test to fill the queue and stack with N values and attempt to print N+1. **What happens? Why do you not need to test for putting too many items into the queue?**