Module 2

Goals-
Build dynamic structures using **pointers**

You will create simple structures using dynamic memory. You will implement a Stack class and a Queue. Each will have a pointer to the appropriate node struct and the designated functions. These are diagrammed at the end of this document.

1. Stack-like behavior. 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 this. You could make your program more difficult. What data member will you need?

You will create a singly linked Stacknode struct. You will need to add an element and to take off an element. You should have functions:

```
void add(Type_of_Data Value)
Type_of_Data remove()
```

Type_of_Data is char for this module. So add() will take char, create a node, put your char in the data part of the node, and adjust pointers as necessary. The remove() function will return the data in the top node. It will also remove the node and adjust pointers.

Demonstrate your stack works by having the user enter a series of characters, put them in the stack, and display them in reverse order.

This is all that is required.

2. Queue-like behavior. 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. Be careful when you research this. You could make your program more difficult. What data members will you need?

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

```
void add(Type_of_Data Value)
Type_of_Data remove()
```

Type_of_Data is char for this module. So add() will take char, will take char, create a node, put your char in the data part of the node, and adjust the back pointer to point to it. The remove() function will return the data in the front node. It will also remove the node and adjust the front pointer.

Demonstrate your queue works by having the user enter a series of characters, put them in the queue, and display them in order to the screen.

NOTE: The only way to display the contents of either data structure is to empty it! You will NOT have another pointer that reads the values and leaves the nodes unchanged.
NOTE: De-allocate memory as appropriate.
HINT: You will use your code as part of assignment 4.

Grading

Programming style- 10%

Create the nodes that use links properly- 110%

Simple first in last out structure- 30%
- Create it
- Correctly implement add()
- Correctly implement remove()

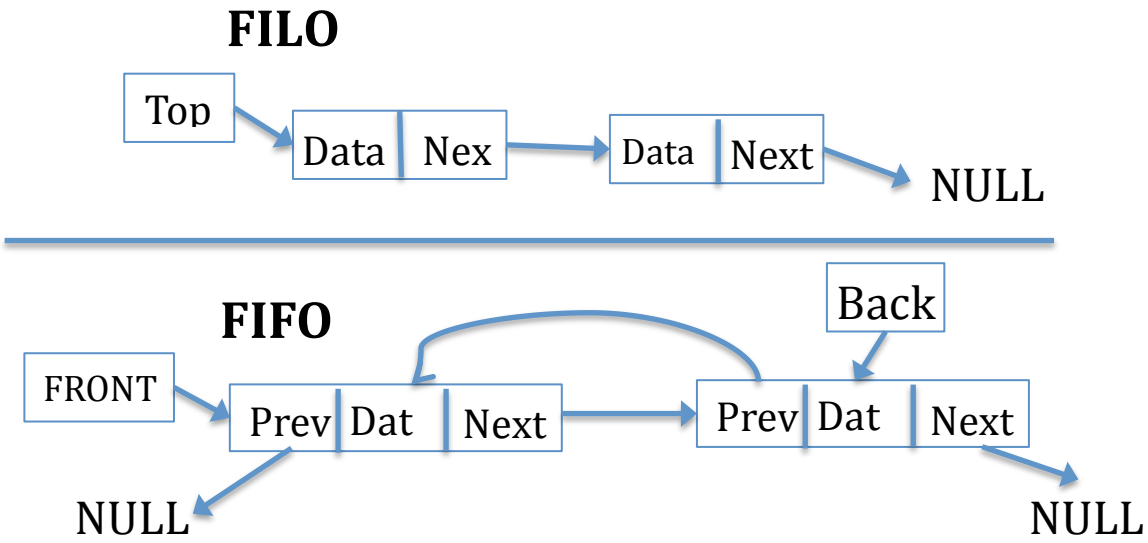
Simple first in first out structure - 30%
- Create it
- Correctly implement add()
- Correctly implement remove()

Proper use of dynamic memory- 10%
--If there serious memory management issues deductions can be made from creating or implementing a structure.

Brief description of how you tested your structures- 10%

Note: Remember that you cannot use or copy code from any source. Your work must be your own.

To help visualize the structures:



With a doubly linked list Front and Back can be at either