

Data Structures and Algorithms

Assignment 4 (Linked List)

Deadline: 23: 55 Hrs, April 9, 2017 **Marks:** 100 **Roll Number:**

—
Implement a class named `LinkedList` which has the following functionalities.

Q1. `IsEmpty()`: Return true if FRONT/TAIL is pointing to NULL otherwise false. (5 points)

Q2. `InsertAtFront (val)`: Creates a new node with data = val and next = front and sets front of the list to the newly created node and checks if tail is pointing to NULL then sets it to front. (5 points)

Q3. `InsertAtTail(val)`: Creates a new node with data = val and next = NULL checks if list is empty. If it is then it sets front = tail = newly created node else tail->next = newly created node. (10 points)

Q4. It has two parts: Let's say you have the following `LinkedList` (7.5+ 7.5 points)

1 => 5 => 6 => 9 => -50 => 75

- **`InsertAfter(val, key)`:** It should enter the new Dabba with value **key**, after the value **val**. e.g for the above linked list if **`InsertAtAfter(9, 35)`** is called then the new list should become:

1 => 5 => 6 => 9 => 35 => -50 => 75

- **`InsertBefore(val, key)`:** It should enter the new Dabba with value **key**, before the value **val**. e.g for the above linked list if **`InsertAtAfter (9, 35)`** is called then the new list should become:

1 => 5 => 6 => 35 => 9 => -50 => 75

Q5. `T GetFront()`: returns the value pointed by FRONT. (5 points)

Note: *This function must be used only if the `IsEmpty()` returns false.*

Q6. `T GetTail()`: returns the value pointed by TAIL.

Note: *This function must be used only if the `IsEmpty()` returns false.* (5 points)

Q7. `Dabba* Search(key)`: returns the pointer to the element **key** if the list contains it. Note: *if the key is not found it should return NULL.*

Q8.

- **`RemoveFront()`:** Stores front of the list in a temp variable, point **FRONT** of list to the next element in the list and deletes temp.

- Also make sure if the list is about to become empty it MUST reset FRONT and TAIL to NULL.
- Make sure you put the Check if the list is already empty then it shouldn't delete anything (rather return immediately).
- **RemoveTail()**: Move Tail to the 2nd last element of the array. And delete the last element in the list. Make sure you put the checks of list being empty(as stated in the above cases).

Q9. Now implement stack using LinkedList data structure.

```
template<class T>
Class Stack{
LinkedList<T> S;
// no other memory then this.
Stack();
void Push(T V);
T Pop();
bool IsEmpty();
};
```

Q10. Implement Queue using LinkedList data structure.

```
template<class T>
class Queue{
LinkedList<T> S;
// no other memory then this.
Queue();
void Enqueue(T V);
T Dequeue();
bool IsEmpty();
};
```

Note: Now with the help of first 8 questions you can easily implement stack & queue.