**BRAC University**

**Department of Computer Science and Engineering**

**CSE 220: Data Structures**

**Task 1**

Implement a **MyLinkedList** ADT that is a singly-linked-list-based list ADT with finite memory.

**Elements**

The elements in **MyLinkedList** are **MyNode** that consists of a generic Object element and a reference to the **next** MyNode.

**Structure**

The **MyLinkedList** container is an **array** named**LinkedListArray** that is an array of instances from the MyNode class. The **MyLinkedList** contains a finite stack named **FreeNodeList.** The **FreeNodeList** contains MyNode objects that are not currently used in the MyLinkedList. The**MyLinkedList** ADT also contains a finite queue named**UsedNodeList** that holds the nodes (instances of MyNode) that are currently in use (i.e., active part of the MyLinedList).

[You are not allowed to use any class variable other than**UsedNodeList, FreeNodeList** and **LinkedListArray**.]

**Constructors and their Helper Method of MyNode Class**

**MyNode (Object Element )**

Pre-condition:

The Object Element is not Null.

Post-condition:

This is a constructor of **MyNode** class. Initializes the Object element and the next pointer to null.

**Constructors and their Helper Method of MyLinkedList**

**MyLinkedList (int size)**

Pre-condition:

The argument size is positive.

Post-condition:

This is another constructor of MyLinkedList class. This constructor allocates an array of MyNodes named LinkedListArray with the defined size and creates the stack FreeNodeList and the queue UsedNodeList.

**Methods of MyLinkedList**

**void insert (Object newElement)**

Pre-condition:

The MyNode newElement is not Null and the FreeNodeList is not empty.

Post-condition:

This method allocates a Object named newElement from the FreeNodeList and inserts (enques) the “newElement” in the**UsedNodeList.**

Instead of using the keyword “new” for space allocation for a new MyNode object, it takes (pops) a MyNode object from the**FreeNodeList** of **MyLinkedList** class. Then, assigns the Element in the MyNode.

Example:

MyNode n = FreeNodeList.pop();

n.Element = “CSE220”;

**MyNode remove ( )**

Pre-condition:

UsedNodeList is not empty.

Post-condition:

This method removes (deques) a MyNode object from the UsedNodeList and returns (push) the MyNode object to the FreeNodeList.

**MyNode search (Object searchElement)**

Pre-condition:

The UsedNodeList is not empty, the searchElement is not null.

Post-condition:

It searches for the searchElement in the UsedNodeList. If found, returns that MyNode object. Otherwise, returns Null.

**void swap (MyNode node1, MyNode node2 )**

Pre-condition:

The two MyNodes node1 and node2 are not null and both are in the UsedNodeList.

Post-condition:

This method swaps node1 and node2.

**void clear ( )**

Pre-condition:

None.

Post-condition:

This method removes (dequeues) all MyNodes from the UsedNodeList and returns (push) them back to the FreeNodeList.

**boolean isFullUsedNodeList ( )**

Pre-condition:

None.

Post-condition:

Returns true if the UsedNodeList is full (i.e., the FreeNodeList is empty). Otherwise, returns false.

**boolean isEmptyUsedNodeList ( )**

Pre-condition:

None.

Post-condition:

Returns true if the UsedNodeList is empty. Otherwise, returns false.

**boolean isFullFreeNodeList ( )**

Pre-condition:

None.

Post-condition:

Returns true if the FreeNodeList is full (i.e., the UsedNodeList is empty). Otherwise, returns false.

**boolean isEmptyFreeNodeList ( )**

Pre-condition:

None.

Post-condition:

Returns true if the FreeNodeList is empty. Otherwise, returns false.

**void showUsedNodeList ( )**

Precondition:

None.

Postcondition:

Outputs the elements of the UsedNodeList. If the UsedNodeList is empty, outputs “Empty list”.