<MyLinkedList> Design

Name: Rahul Jayaraman Period: 3

**Description:**

This class implements a doubly linked list. This consists of many nodes with pointers to the previous node as well as the next node; because there are two pointers from a node to another, it is referred to as doubly linked. This is useful if you want to traverse a list from the first to the last element, or in the opposite direction too – something that is impossible with a singly linked list. A MyLinkedList object can store data that must be attached to/hold references to another data element – especially if the elements need to be kept in a specific order.

**Services:**

The following services are available:

int size()- This method returns the size of a MyLinkedList object, stored in the myLinkedListSize state variable, changed whenever we add a method.

private getIndex() – This helper method will get the index of a particular object in a linked list; it will prove useful in some of the methods that we will implement.

boolean add(E obj)- This method appends a value/object to the end of a given LinkedList. This simply uses the setNext() method to create a new ListNode with the desired value; it then reassigns the pointer last to this new ListNode and sets the previous pointer to the ListNode that was previously last using the setPrev method – this is O(1).

void add(int index, E obj)- This method will add an object at the desired index and then shift all the objects to its right one index down. We will traverse the linked list and perform this action through a series of pointer assignments. If the object is further down the linked list, it will take a longer time, making it run in O(n).

E get (int index) – This method gets the desired value at a specified index. Once again, we must traverse the linked list in order to get to this index. We can then just use the .getValue() method to get this value. Runs in O(n) time.

E set (int index, E obj) – This method changes a given index’s value to the value that is input into the argument. Once again, we must traverse the linked list and then use the set method once we reach the desired index, using the .setValue() method. Runs in O(n) time.

E remove (int index) – This removes the value at a given index and shifts all the values after that back one. What we can do is just a little bit of pointer reassignment at the index. Runs in O(n) time.

Iterator<E> iterator() – Returns null; will be developed in a later lab.

ListIterator<E> listIterator() – Returns null; will be developed in a later lab.

**Internal Data Structures and State**

When the DoubleNode constructor is initialized, a new DoubleNode is created. There are no data structures such as arrays; it is just a bunch of interlinked DoubleNodes. The state variable for size, myLinkedListSize, will store the instance variable and will be set in the constructor. There are also some other state variables – first and last, with pointers to the first and last node. These actually enable us to *access* the linked list and its contents.