Project

Create in C++ a template class MyLinkedList<T>. The objects of the type MyLinkedList<T> are collections of objects of type (class) Node<T>, where each node points to the next node and to the previous node.

1. The class Node<T> must have a member "data" of the type T and two pointers of type Node<T>*. The class Node<T> must also have a private constructor with signature:

```
Node(T v)
```

which create a node with data = v and the pointers with null value. The class Node<T> must declare the classes MyLinkedList<T> and MyLinkedListIterator<T> to be friend.

2. The class MyLinkedList<T> must have two private members, one "first", for the head of the list (first node) and one "last" for the last node in the list. The class MyLinkedList<T> must have 3 public constructors with the following signatures:

```
MyLinkedList()              // empty list

MyLinkedList(T v)           // list of a single node, whose member "data" is v.

MyLinkedList(T v [])        // list whose nodes correspond to the elements of the array v
```

It must also have the following dynamical methods:

```
public Vector<T> toVector()  // return the vector of objects of type T corresponding to
                             //   MyLinkedList.

public MyLinkedList<T> copy()     // returns a copy (a clone) of the MyLinkedList.
```

```
public T head()                  // return the value of "data" of the first node.
public MyLinkedList<T> tail()    // return the MyLinkedList without its first node.
public T end()                   // return the value of "data" of the last node.
public void append (T e)         // creates a node with "data"=e at the end of the MyLinkedList.
public void concat (MyLinkedList<T> lc)   // concatenates lc at the end of MyLinkedList.
public MyLinkedListIterator<T> iterator()      // method to initialise the iterator
                                               //   MyLinkedListIterator for the current
                                               //   MyLinkedList.
```

3. The class MyLinkedListIterator<T> must have three private members (pointers), for instance:

```
private Node<T>*  itnext;              // next node
private Node<T>* current=null;         // current node
private MyLinkedList<T>* ch;           // MyLinkedList object upon which we iterate.
```

The class MyLinkedListIterator<T> must also have a constructor with signature:

```
public MyLinkedListIterator(MyLinkedList<T>  a)
```

and the dynamic methods:

```
public boolean hasNext()
public T next()
public boolean hasPrevious()
```

```
public T previous()

public goToBegin()          // sets the iterator to the begin of the list.

public goToEnd()            // sets the iterator to the end of the list.

public void set (T v )      // assign v as the value of "data" in the node that is at the
                            // current position.

public void add (T v )      // adds a node ("data" = v) after the current position.

public void remove()        // removes the node that is at current position.
```

4. The classes MyLinkedList<T> and MyLinkedListIterator<T> must use one exception class MyLLInvalidAccessException, extension (public) of exception. It must receive one numeric parameter to identify (at least) **six** cases of exception:

```
case 1: msg= "Invalid head() call: empty list";

case 2: msg= "Invalid end() call: empty list";

case 11: msg= "Invalid next() call: hasNext() false";

case 12: msg = "Invalid previous() call: hasPrevious() false";

case 13: msg = "Invalid set(T v) call: undefined current position";

case 14: msg = "Invalid remove() call: undefined current position";
```

Attention: You must also write a main C++ program to check if your classes work properly (informal test).