EE 441 HOMEWORK #2

Linked Lists

Due: December 3, 2018, 23:59
For questions: hdoga@metu.edu.tr

Part 1:

You are required to design a `LinkedList` class which allows you declare objects of type `LinkedList<T>`. The class provides the interface to the linked list as a linear storage of items of type `T` by hiding the details of the pointer operations.

The declaration and partial implementation of the `LinkedList` class is in ”`link.h`”, which includes the `Node` class in the given ”`node.h`”.

Implement all member functions declared according to the comments and the visual examples below. Some of the member functions are implemented for you. Please check them carefully to see how to update the member variables to maintain the state of the linked list object.

All `Insert__` methods take the data of the node as argument, dynamically create the node using `GetNode` and insert the node in the linked list.

`currPtr` points at the newly inserted node after all insertion operations.

All `Delete__` methods return the dynamically allocated memory of the deleted node. `currPtr` points at the node that was following the deleted node. If the last node is deleted, `currPtr` becomes `NULL`.

All methods should maintain the state of the `LinkedList` object by correctly modifying the class members.

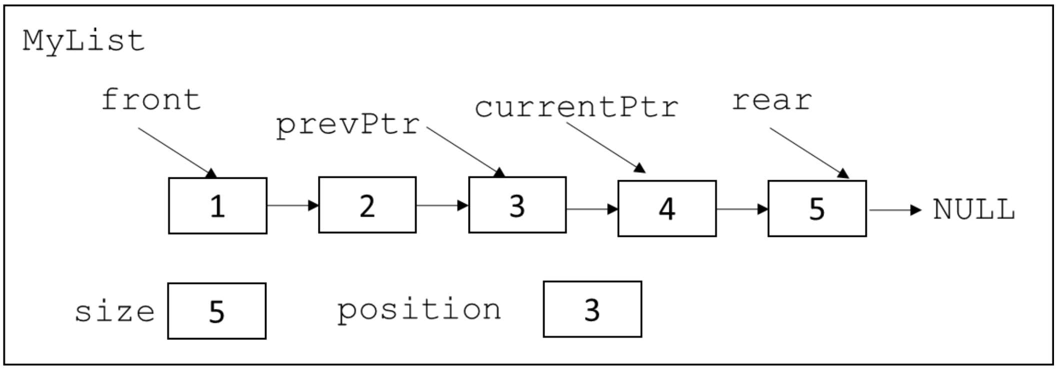
You can reuse member functions of the class (such as `InsertAt` and `DeleteAt`) to implement other member class member functions.

Please make sure that you make the necessary checks to prevent any undesired state of the `LinkedList` objects.

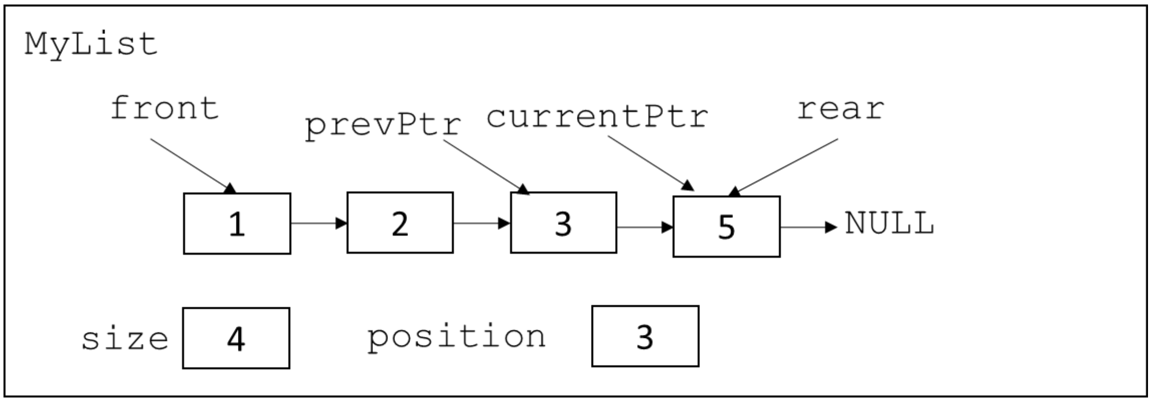
Note that `LinkedList` is a class template. The examples below are for a `LinkedList<int>` class. Your code will be checked using other template parameters (i.e. for different `T`s, not necessarily primitive data types) as well.

Examples:

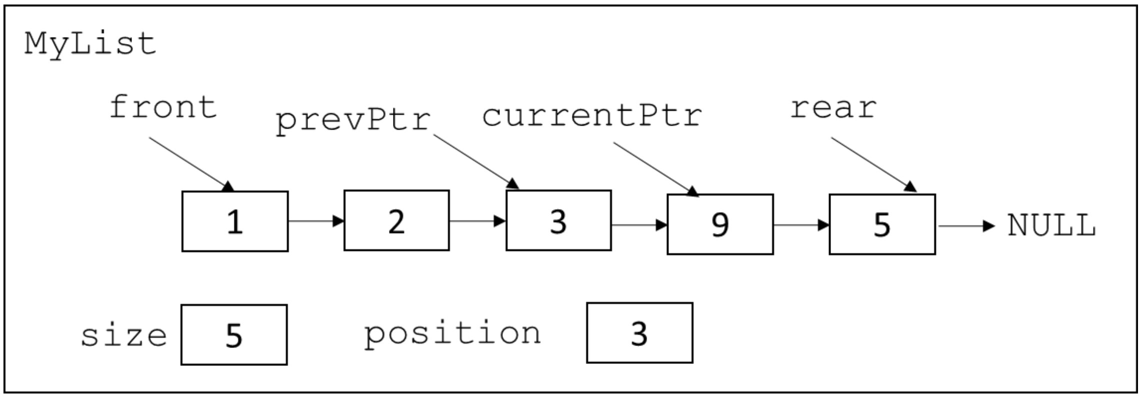
```
LinkedList<int> MyList;

for(int i=1;i<6;i++)

    MyList.InsertAfter(i);

MyList.Reset(3);
```



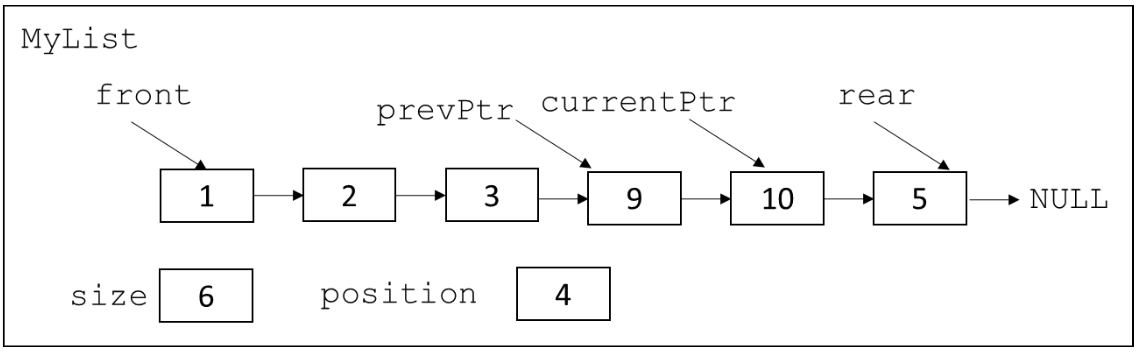
```
for(MyList.Reset();!MyList.EndOfList();MyList.Next())

    cout<<MyList.Data()<<" ";

//prints 1 2 3 4 5.

//currPtr is NULL because of the list iteration in the loop

MyList.Reset(3);

MyList.DeleteAt();
```



`MyList.InsertAt(9);`



`MyList.InsertAfter(10);`



Part 2:

Implement the `PriorityQueue` class using the `LinkedList` that you implemented. The declaration and partial implementation of the `PriorityQueue` class is in "`PQ.h`".

The `PriorityQueue` maintains an ordered list where the highest priority item is at the front position. Assume that the priority queue stores items of type `T`. The `<`, `>` and `==` operators are defined or overloaded for `T` such that if `item1` has higher priority than `item2` then `item1<item2` and if `item1` has the same priority as `item2` then `item1==item2`.

When a new item is inserted in the `PriorityQueue`, `QInsert` member function searches for the correct location to insert, such that the order is maintained.

If there exist items with the same priority as the new item, the new item must be inserted after these items.

Note that `PriorityQueue` is a class template. The examples below are for a `PriorityQueue <int>` class. Your code will be checked using other template parameters (i.e. for different `T`s, not necessarily primitive data types) as well.

Examples:

Current PQ contents for

```
PriorityQueue<int>MyQ;
```

10	20	30	40

```
MyQ.Qinsert(25);//changes the queue as follows
```

10	20	25	30	40

```
int most_important=MyQ.QDelete();

cout<<most_important;

//prints 10 on the screen and changes the queue as follows.
```

20	25	30	40

Regulations:

1. Use **Code::Blocks IDE** and choose GNU GCC Compiler while creating your project. Name your project as "e<student_ID>_HW1". Send the whole project folder compressed in a rar or zip file. You will not get full credit if you fail to submit your project folder as required.

2. The code you uploaded should be compilable and the built file should be executable. **We have to see some output to grade your homeworks.** So please make sure that you have uploaded a working version for your homework.

3. You should insert comments to your source code at appropriate places without including any unnecessary detail. Comments will be graded (On the condition that you have managed to send a working code). A code with insufficient/excessive comments is not a proper piece of work.

4. The homework is to be prepared individually. It is not allowed to prepare the homework as a group. METU honor code is essential. Do not share your code. **Any kind of involvement in cheating will result in a zero grade for all homeworks, for both givers and receivers.**

5. You have to give the links to any website you have benefitted from. You can add them to your code as comments in the beginning.

6. Late submissions are welcome, but penalized according to the following policy:

    - 1 day late submission: HW will be evaluated out of 70.
    - 2 days late submission: HW will be evaluated out of 50.
    - 3 days late submission: HW will be evaluated out of 30.
    - 4 or more days late submission: HW will not be evaluated.

Good Luck!