CMPT225, Spring 2023
Homework Assignment 2
Due date: Friday, February 10, 2023, 23:59

You need to implement the following classes:
Part 1:
- *assignment2.MyLinkedList.java*

Part 2:
- *assignment2.MyQueueOperations.java*

Note that all files must be in under the package (folder) *assignment2***.**
You may add more classes to your solution if necessary.

Submitting your solution: Submit all your files in **assignment2.zip** to CourSys.
Make sure your zip file can be unzipped using the command "unzip assignmen2.zip" in CSIL.
The zip file must contain exactly one folder, named *src*.
In the *src* folder there is a folder called *assignment2*.
The folder *assignment2* will contain all the .java files.

Discussion with others: You may discuss the assignment with your classmates/tutors (or anyone else), but coding must be entirely your own.

References: You may use textbooks, wiki, stack overflow, geeksforgeeks, etc. If you do, specify the references in comments. Posting questions online asking for solutions (e.g. using chegg.com) is prohibited.

Readability: Your code should be readable using the standard Java conventions. Add comments wherever is necessary. If needed, write helper functions or add classes to improve readability.

Compilation: Your code MUST compile in CSIL using javac.
Make sure that your code compiles without warnings/errors.
If the code does not compile in CSIL the grade on that part will be 0 (zero).
Even if you can't solve a problem completely, make sure it compiles.

The assignment will be graded mostly **automatically**, with some exceptions.

Do not add main() to your solutions. The main() method will be in the test files.

Warnings: Warnings during compilation will reduce points.
More importantly, they indicate that something is probably wrong with the code.

Testing: Test your code. Examples of tests are included. Your code will be tested using the provided tests as well as additional tests. You should create more tests to check your solution.

Good luck!

Part 1 [50 points] - class MyLinkedList<T>

In this part you need to implement a class MyLinkedList. This is a generic class representing a *list of objects*. The operations on the list are as follows:

- adding and removing elements from the left and from the right.
- reversing the list
- obtaining the middle element
- getting the size of the list

All operations must run in O(1) time.

Specifically, you need to implement the following constructors and operations

`public MyLinkedList()`
Creates an empty linked list

`public void addLeft(T item)`
Adds the new item to the left of the list.

`public void addRight(T item)`
Adds the new item to the right of the list.

`public T removeLeft()`
Removes the leftmost item from the list and returns it.
If the list is empty, throws ***NoSuchElementException***.

`public T removeRight()`
Removes the rightmost item from the list and returns it.
If the list is empty, throws ***NoSuchElementException***.

`public void reverse()`
Reverses the list.
For example, if the list was 1-2-3-1-4, after running the method, it becomes 4-1-3-2-1.

`public T getMiddle()`
Returns the item in the middle of the list.
If the list has even length (and there are two middle nodes) the method returns the node closer to the right end.
For example:

- if the list is 3-4-5-6-5-4-3, the method returns 6.
- if the list is 100, the method returns 100.
- if the list is 0-6-8-100, the method returns 8.
- if the list is 1-2, the method returns 2.
- If the list is empty, throws ***NoSuchElementException***.

`public int size()`
Returns the size of the list

`public boolean isEmpty()`
Checks if the list is empty.

Part 2 [50 points] - class MyQueueOperations<T>

The interface *Queue<T>* declares the standard operations: enqueue(), dequeue(), isEmpty().
You have two concrete implementations: QueueArrayBased.java and QueueLinkedListBased.java

Your goal is to implement the class MyQueueOperations. The class has several public static methods manipulating Queue. The public methods are as follows.

```
public static <T> int size(Queue<T> q)
```
Returns the number of elements in q.

```
public static <T> Queue<T> clone(Queue<T> orig)
```
Returns a copy of orig. The items are copied from orig to the new queue using = operator. For the concrete type of the returned object, you may use either *QueueArrayBased* or *QueueLinkedListBased*. That's up to you.

```
public static <T> void reverse(Queue<T> q)
```
Reverses the order of the elements in q.

```
public static <T> boolean areEqual(Queue<T> q1, Queue<T> q2)
```
Checks if the two queues have the same items in the same order.
The items in the queues are to be compared using == operator.

After the methods return, the input queues must be in the same state as in the beginning.