COMP 229
Exam 1 (Total: 100 points)

- There are in total 3 programming problems + 1 extra credit problem. You should be able to answer all problems in one java file.
- You may not work with others, look things up on the internet nor use lecture notes or homework during the exam period.
- **Fail to do any of the above may result in zero.**

Use LinkedNode.java, LinkedList.java, Stack.java and Queue.java provided, **pick 3 of the following methods to implement**. You may choose any 3 of the following options.

For each method, you need to:

1. Explain the general algorithms in plain English in the comments above, including the special cases. (5 pts)
2. Implement the method according to the requirements. (20 pts)
3. And report the runtime complexity in the Big-O format with brief explanation in the comments above. (5 pts)

The remaining 10 pts will be graded for the efficiency and the style of your code. For example, if your method has a runtime complexity of O(n2) - n square but the ideal solution has a runtime of O(n), you may lose some points. If your code is messy without proper indentation, you may lose some points.

Hints:

- You may want to utilize the idea of Stack and Queue when implementing some of the methods, even though they are all dealing with linked lists.
- Make sure you handle special cases carefully, e.g. when the list is empty, etc. and write proper test cases to test your methods.
- Make sure you test your methods thoroughly.

Option A:

```
public static LinkedList evenBeforeOdd(LinkedList ls)
```

Write a method called `evenBeforeOdd` that takes a linked list of integers and returns a new linked list in the way that all even numbers appear before all the odd numbers in the new linked list.

```
Input: 1 2 3 4 5 6 7
Output: 2 4 6 1 3 5 7
```

Option B:

```
public static LinkedList reverseList(LinkedList ls)
```

Write a method called `reverseList` that takes a linked list of integers and returns the linked list in a reversed order.

.

```
Input: 1 2 3
Output: 3 2 1
```

Option C:

```
public static LinkedList evenIndicesBeforeOdd(LinkedList
ls)
```

Write a method called `evenIndicesBeforeOdd` that takes a linked list of integers and returns a new linked list in the way that all nodes with even indices appear before all nodes with odd indices.

```
Input: 1 8 29 20 4 6 39
Output: 1 29 4 39 8 20 6
```

Option D:

```
public static LinkedList combineList(LinkedList ls1,
LinkedList ls2)
```

Write a method called `combineList` that takes 2 sorted linked lists of integers of the same size, merges them into one sorted linked list and returns the one sorted linked list.

```
Input: 1 2 3 8 and 1 8 20 31
Output: 1 1 2 3 8 8 20 31
```

Option E:

```
public static boolean isPalindrome(LinkedList ls)
```

Write a method called `isPalindrome` that takes a linked list of integers and returns true if the linked list is palindrome.

```
Input: 1 2 3 2 1
Output: true
```

Option F:

```
public static LinkedList sortList(LinkedList ls)
```

Write a method called `sortList` that takes a linked list with each node containing either 0, 1 or 2 and returns the linked list in a sorted order.

```
Input : 1 1 2 0 2 0 1 0
Output : 0 0 0 1 1 1 2 2
```

Extra Credit: Given a queue, how do we sort it in an ascending order? (5 pts)

Write a method called `sortingQueue` that takes a queue of integers, and return a sorted queue in an ascending order.

```
public static Queue sortingQueue (Queue q)
```