Assignment:

Inspired by Q1, write a program to check if a piece of text that the user enters is a palindrome. In this case, you add each character to both a stack and a queue as it is read.

After each character has been put onto both the stack and the queue, start removing characters from both the stack and the queue, and comparing them. If the word is a palindrome, the first character popped will be the same as the first character dequeued, and so on for the second and subsequent characters until the stack and queue are empty.

Name: Peter Conway

Student ID: 15498698

Degree: 2nd BScIT

Module: CT2109

Submitted: 13th March

Statement:

Design implement and test a Java program as specified in Assignment 4.

Analysis and Algorithm:

- Create an empty Stack and Queue
- Read user input into a String s
- Convert string to lower case
- Incrementally push s into the Stack
- Enqueue s into the linked list Queue
- Incrementally pop elements out of the Stack into a String popped
- Dequeue into a String dequeue
- Incrementally check equality between each element of the String's popped and dequeue
- For each equality increment a counter
- If the counter equals the length of the String then change the Boolean counter to true to indicate the String is a Palindrome
- Print out the results

Implementation:

```
import java.util.*;

public class PalindromeCalc {
	public static void main(String[] args) {

	boolean isPalin = false; // basic switch for palindrome authenticity
	int Pcounter = 0;//counts number of equalities in strings
	String popped = ""; // read from stack
	String dequeue = ""; // read from queue

	Stack st = new Stack();
	LLQueue list = new LLQueue();

	System.out.println("Enter: ");
	Scanner scan = new Scanner(System.in);// scan string to be checked
	String s = scan.next();

	for(int i = 0; i<s.length(); i++) {
		st.push(s.charAt(i));// add string to stack

	}

	list.enqueue(s); // add string to queue


	for(int i = 0; i<s.length(); i++) {
	popped = popped+st.pop();// pop from stack and add to string

	}

	dequeue = (String)list.dequeue(); //dequeue into a string

	for(int i = 0; i<s.length(); i++) {
		if (dequeue.charAt(i)==popped.charAt(i))// check equality
against each character
			Pcounter++; // increment if equal
	}

	if(Pcounter==popped.length())// compare number of equalities to the
length of the string
		isPalin = true;
	System.out.println(dequeue+"     ");
	System.out.println(popped+"            "); // print stuff out
	System.out.println("The String is a Palindrome:" + isPalin);
	}
}
```

Also attached via .zip is the files for the LLQueue, node and queue

Testing:

Testing various input Strings.

- Test: “bacon”
- Expected: False
- Result: As expected

- Test: “BaKAb”
- Expected: True
- Result: As expected

- Test: “1234554321”
- Expected True
- Result As expected

From these test results I can see that for most generic combinations of strings, that don’t exceed a certain character count, this method will prove substantial.