Assignment 4
Due October 14

PROGRAM 1

A Palindrome is a string that reads the same forwards and backwards.
For example Madam I'm Adam is a palindrome (You ignore punctuation, spaces, and upper/lower case).

Write a program that determines whether or not a string is a palindrome.

Your program should have a method

```
public boolean is Palindrome(String s)
```

And use only a stack and a queue to determine whether or not a string is a palindrome. Nothing else. Use a stack and a queue of Character.

Use this main

```
public static void main(String[] args)
{

  Scanner input = new Scanner(System.in):
  System.out.print("Enter a string -- end with "ABC");
  String s = input.nextLine();
   while (!s.equals("ABC")
   {
      boolean pali = isPalindrome(s);
      if (pali)
         System.out.println(s + "is a palindrome");
      else
         System.out.println(s + "is not  a palindrome");

       System.out.print("Enter a string :");
    }
}
```

Test your program the following. I made all data upper case but spaces and punctuation should not be included.

- DON'T NOD (Here you are testing DONTNOD)
- I DID, DID I? (Here you are testing IDIDDIDI)
- MY GYM
- RACECARER
- SOLOS
- RED RUM, SIR IS MURDER
- STEP ON NO PETS
- STOP AND TOP
- TOP SPOT
- WAS IT A CAT I SAW?
- EVA, CAN I SEE BEES IN A CAVE?
- NO LEMON NO MEL

PROGRAM 2.

Write a program to determine all primes from 2 to n inclusive.

Your program should use **2 queues**. Here is the algorithm:

Prompt for a number n, such as 100
add all the numbers from 2 to n inclusive to queue1
create another (empty) queue , queue2, to hold the primes
do
    remove the first number p from queue1 and add it to queue2
    iterate through queue1 removing all multiples of p and re-inserting all other numbers back into queue1
while (p< Math.sqrt(n)

Everything left in queue1 is prime so add these to queue2
print queue2, tose are the primes .

DO NOT DO ALL Of THIS IN MAIN.

Program 3.

You have two sorted linked lists of Integer. Merge them into third list.

We did merging with arrays (when we did mergesort) now you can merge two linked lists.

To do this make a new class MergeLists that extends the LList class.
Since MergeLists extends Llist you can uses any of its methods.

Below is a skelton of the program with the constructor that builds the initial two lists.
Fill in the merge() method and the display() method.
I have also attached the LList class to the email.

```
import java.util.*;
public class MergeLists extends LList<Integer>
{
  private LList<Integer> list1, list2, mergedList;
  public MergeLists()  // constructor
 {
   list1 = new LList<Integer>();
   list2 = new LList<Integer>();
   mergedList = new LList<Integer>();
   mergedList.add(0); // a dummy first node
   list1.add(2);
   list1.add(3);
   list1.add(5);
   list1.add(8);
   list1.add(13);
   list1.add(27);
   list1.add(60);
   list2.add(4);
   list2.add(9);
   list2.add(11);
   list2.add(12);
   list2.add(15);
   merge();
 }
```

```
  public void merge()
  {
  // merges list1 and list2 into mergedList
  }

  public void display()
  {
     //prints the contents of lmergedList
  }

 public static void main(String[] args)
 {
   MergeLists m = new MergeLists();
   m.display();
 }
}
```

If you look at the merge we did for mergesort the idea is exactly the same. Now we are going through linked lists rather than arrays.