Assignment 2: Double-base Palindromes and Complexity Analysis

A palindrome is a sequence that reads the same backwards as forwards.

For example: 86668 and 100111001 are examples of a decimal and binary palindrome, respectively.

For this assignment, you are going to write an application which tests if a sequence of numbers is a palindrome or not. Specifically, you are going to write three different methods (with meaningful names) which take a String as a parameter and returns a Boolean which represents whether or not the String is a palindrome:

1. **Method One**:
    a. You should reverse all the characters in the String using a loop and then compare the reversed String to the original to determine if it is a palindrome.
    b. Return either true or false depending on the comparison.
2. **Method Two**:
    a. In this method we are going to compare the characters on an element by element basis using a loop. We will compare the first element to the last, the second element to the second last and so on..
    b. If *at any point* there is no match, we will immediately stop and return false. If we make it through the full sequence and they all match, then we will return true.
3. **Method Three**:
    a. In this method, we are going to use the ArrayStack and ArrayQueue implementations from Blackboard.
    b. Add each character (digit) to both a stack and a queue as it is read.
    c. After all characters have 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.
    d. If there is a mismatch at some point, return false. Otherwise return true.

You are going to test these methods using both decimal and binary numbers. For this reason, you will need to create a utility method which converts a decimal number into its equivalent binary representation and then returns it.

- This method will have one parameter which will be a String representation of the decimal number.
- It will return the binary representation of the number as a String variable.

Testing

We are going to test how efficient each of the three different methods are at identifying palindromes. In order to do this, we are going to use a loop and test them on the numbers (both decimal and binary) from 0 to 1000000. This will involve calculating, in milliseconds, how long it takes to count the number of palindromes in the range (0-1000000) for each of the methods for decimal and binary numbers. For this you can use `System.currentTimeMillis();` to read the time before and the time after.

You should also carry out a test to count the number of instances in which **both** the decimal number **and** its binary equivalent are recorded as palindromes.

> For instance, the decimal number 585 is 1001001001 in binary. We can see here that the number is palindromic in both bases. We want to count how many times this happens in the first 1 million numbers.

Finally, we are going to count the number of primitive operations for each method. To do this, you can create global variables (one for each method) that can be incremented every time an operation occurs (remember to reset them to 0 if you are automating multiple runs).

You should try and figure out how many operations it might take each method before writing any code.

Graphing the results

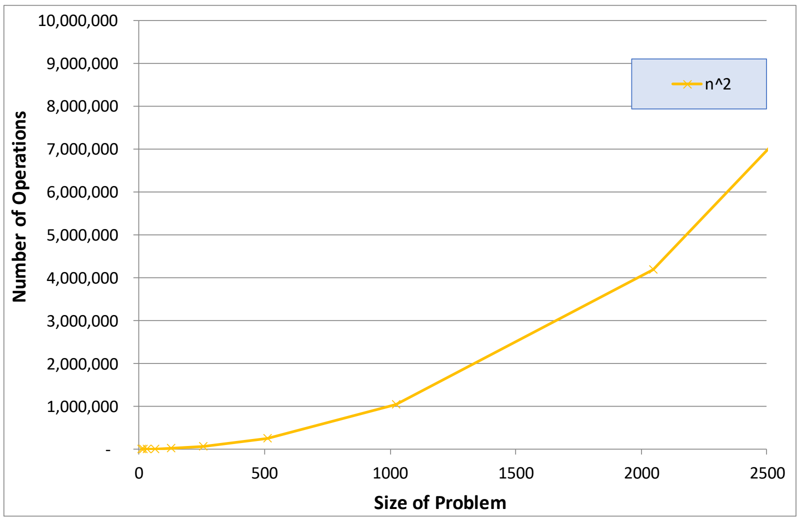
You should plot the number of operations against the range of numbers checked for each method so that you can visually analyse the complexity as the numbers increase. You should carefully consider what kind of intervals to use when generating your graphs in order to draw meaningful conclusions. The y-axis will represent the number of operations required and the x-axis will represent the upper bound of the range checked.

For instance, you could increase the range by 50,000 for each point on the graph:

0 50000 100000 150000 200000 etc…

Recall from class, our analysis of complexity functions. The following graph is for n^2

You will produce graphs like this, but each line of the graph will correspond to a particular palindrome method that you have developed. Size of the problem in your case corresponds to the range of numbers that you are checking.



Note: There is an <u>increase in marks for testing</u> in this assignment due to the complexity analysis.

Example Skeleton Code Outline

```
public class AssignmentTwo

{
        //Declare any global variables required (e.g. operation counts for each method)

        //Main Method
        public static void main (String[] args)
        {
                //Declare any variables used (e.g. for timing etc.)

                //Test each method (looping over the binary/decimal numbers for each)
                //by calling your defined methods below

                //Display results for each method

                //Note: Think carefully about the design of your main method
                //If designed correctly, you will be able to automate the running of
                //experiments over many number ranges instead of having to manually
                //change the values for each run. The data produced can then be used for
                //graphing in Excel.
        }

        //Static method for: Palindrome Method 1 (give it a names based on how it works)
        //Takes a String as a parameter and return a Boolean value

        //Static method for: Palindrome Method 2 (give it a names based on how it works)
        //Takes a String as a parameter and return a Boolean value

        //Static method for: Palindrome Method 3 (give it a names based on how it works)
        //Takes a String as a parameter and return a Boolean value

        //Static method for: Converting a decimal number into its equivalent binary representation
        //Takes a String representation of a number as a parameter and return a String value

}
```

Alternatively, you could create a Java interface called Palindrome and have each of your "methods" as separate classes that implement it. You would then create instances of these in your main method as required.

Submission Notes:

- You should submit a *single PDF document* with the following sections:
  - Problem Statement (3 Marks)
    - Describe the problem. You can carry out your own research to describe the overall problem in sufficient detail to demonstrate that you fully understand it.
  - Analysis and Design Notes (5 Marks)
    - Before you begin coding, you should analyse the overall problem and create design notes for yourself. This can include identifying methods that will be required, writing basic pseudocode and outlining the flow of control for the program. You can then use this as a guide when you begin programming.
  - Code (12 Marks)
    - **IMPORTANT**: You must copy and paste your code as text into this section
    - **If you submit a screenshot of the code you will receive 0 marks.**
    - Your code must also contain plenty of *meaningful* comments to fully describe the functionality of each part.
  - Testing (10 Marks)
    - You should extensively test each aspect of the code and provide screenshots of the testing output. You can use Excel to produce the graphs for the increase in the number of operations as the range of numbers checked increases. Each method should be tested. You can plot them together on one graph if you wish. You should provide an analysis of your resulting graphs.

- You have three lab sessions to complete this assignment

Due date: **Friday the 1st of March**