CSE 271 Lab 10 – Recursion
Spring 2022
Assigned: 4/7/2022
Due: 4/10/2022

Introduction:

In this lab, we will solve multiple recursion problems. While the amount of coding might be shorter than previous labs, this will require more thinking and planning rather than raw coding. Start by creating a new project in your Eclipse IDE called **Lab10** and make a new class called **Recursion.java** that has the following static methods:

1. **public static int power(int x, int n)**
2. **public static int sumDigits(int n)**
3. **public static void printBackwards(String word)**
4. **public static boolean isPalindrome(String word)**
5. **public static int sumPositive(int[] array)**
6. **public static int max(int[] array)**

Problem 1 - Power:

The power method of a positive integer recursively computes the term x^n using the following formula:

$$x^n = \begin{cases} 1 & if\ n = 0 \\ x * x^{n-1} & if\ n > 0 \end{cases}$$

For example:

- $3^2 = 9$
- $2^3 = 8$
- $100^0 = 1$

Problem 2 - Sum of the Digits of an Integer:

In this problem, you need to find the sum of the digits in a positive integer. That is, if an integer consists of *n* digits, then you need to sum all of these digits. For example:

- The integer 351 has 3 digits of 3, 5, and 1. The sum of is 3 + 5 + 1 = 9
- The integer 1137 has 4 digits of 1, 1, 3, and 7. The sum is 1 + 1 + 3 + 7 = 12

The idea is that if we can separate one digit at a time and add it then we can break our problem into a smaller piece per recursive call. Specifically, if we want to compute sumDigits(259), then we can do the following:

- sumDigits(259) = sumDigits(25) + 9
  = sumDigits(2) + 5 + 9 [sumDigits(25) = sumDigits(2) + 5]
  = 2 + 5 + 9 [sumDigits(2) = 2]
  = 16

Recall the similar problem presented in the lecture where we utilized the modulus operator to extract the last digit from the integer. For example, 552 % 10 = 2. You can also use integer division to separate the remaining digits from the last digit. For example, 552 / 10 = 55.

Problem 3 – Printing a String Backwards:

When presented with a String, you need to print it backwards (in reverse). That is, if we pass the recursive method the String "lake", then it should print out "ekal". To do this recursively, use the following notes while designing the method:

- If the String contains any characters (i.e., it is not an empty String)
    - Print the last character in the String
    - Print the String backwards without the last character

Problem 4 - Palindromes:

A palindrome is a String that is the same forward and backwards as well as case insensitive. In the first project, you created a program that uses a loop to determine whether a String is a palindrome or not. However, it is also possible to do recursively. Use the following notes to assist you in designing this recursive method.

- A String containing fewer than 2 letters is always a palindrome
- A String containing 2 or more letters is a palindrome if
    - The first and last letters are the same
    - The rest of the String (without the first and last letters) is also a palindrome

Recall that for a String word in Java:

- word.length() returns the number of characters in the String
- word.charAt(i) returns the character at index i
- word.substring(i, j) returns the substring of word that starts at the index I and ends with index j

So, if the word is "happy", then the word.length() is 5, word.charAt(1) is 'a', and word.substring(2, 4) is "pp".

Problem 5 - Sum the Positive Values in an Array:

Given an array of integers, find the sum of all positive values in the array. If we have an array of [7, 3, -6, 4, -9], then our recursive method should return 14.

Problem 6 – Find the Maximum Value in an Array:

Given an array of integers, you need to find and return the maximum value recursively. The array will always have at least one item. If we have an array of [7, 3, 11, 5, 10], then our recursive method needs to return 11. This idea is to separate the first number to reduce the array to a smaller one and then find the max in the smaller array. Following, we compare the separated number with the max of the sub-array which in turn recursively finds the overall maximum value. For example:

- Max in [5, 3, 11] compares 5 and Max in [3, 11]
    - Max in [3, 11] compares 3 and Max in [11]

        - Max in [11] is 11 as it only has one element
    - Now, compare 3 and 11. The Max is 11
- Now, compare 5 and 11. The Max is 11

JUnit Testing:

Write a JUnit test class called **RecursionTester.java** to test the **Recursion.java** class and all of the recursive methods. When you test, you should make sure that you include edge cases. You can call static methods using Recursion.power(3, 2) for example. You do not need to create an object to call static methods. You do not need to write assert statements for the methods which have a return type of void. Just call the method and check console output for correctness.

Additional Notes:

- You can use Arrays.copyOfRange(array, start, end), which makes a new sub-array from "array" that contains the elements between, and including, start to end. This will be useful for the array recursion problems. Please review the documentation for this method.
- You may use helper methods if you wish. You do **NOT** need to create JUnit tests for both the helper method and the recursive method being utilized by the helper method. Only create JUnit tests for the helper method.
- All problems must be solved using recursion. If loops are seen within any of the methods, you will receive a 0 for that problem.
- You may **NOT** have any instance properties in the **Recursion.java** class.
- Make sure you include JavaDoc comments for all methods and the class including parameter and return descriptions.
- Make sure that you name the classes correctly.

Submission Instructions:

After you have completed the lab assignment, locate your source code (**Recursion.java and RecursionTester.java**) in your workspace and submit it to the corresponding Lab 10 assignment's CODE plugin.

Rubric:

Task	Grade
Recursion	
Power	15
Sum of the Digits of an Integer	15
Printing a String Backwards	15
Palindromes	15
Sum the Positive Values in an Array	15
Find the Maximum Value in an Array	15
JUnit Tests	10
Adequate JavaDoc included and followed the Miami University coding guidelines (can only lose points)	0 (-10)
Total	**100**