Lab Assignment 10, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University
Recursion

Lab Overview:

In this lab, you will practice recursion. That is solving problems by writing recursive methods where a problem needs to be broken into smaller subproblem(s). Write a class named **Recursion** that has the following static methods:

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

Important Note:

- You must use recursion for solving all the problems. If you use loops for any of the problems, you will get zero for that problem.
- You cannot have any instance variables or constants in the Recursion class.

Problem 1: Factorial

The factorial of a positive integer n, denoted by $n!$, is the product of all positive integers less than or equal to n. For example,

- Factorial of 3 is 3! = 3 * 2 * 1 = 6
- Factorial of 5 is 5! = 5 * 4 * 3 * 2 * 1 = 120
- Factorial of 0 is 0! = 1

Mathematically, we can define the factorial in the following way,

$$n! = \begin{cases} 1 & if\ n \leq 1 \\ n * (n-1) * (n-2) * \ldots * \ 2 * 1 & if\ n \geq 2 \end{cases}$$

This problem is similar to what we did in the lecture, where we computed sum of all the numbers from 1 to n. In this problem, we have to find product of all the numbers from 1 to n.

Problem 2: Power

The power method 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,

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

Lab Assignment 10, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University
Recursion

Problem 3: Sum of digits of an integer

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

- The integer 324 has 3 digits 3, 2, 4. The sum is 3 + 2 + 4 = 9.
- The integer 1025 has 4 digits 1, 0, 2, 5. The sum is 1 + 0 + 2 + 5 = 8.

This can be solved in a recursive way. 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 one and solve it. If we want to compute *sumDigit(324)* then we can do the following:

```
sumDigit(324) = sumDigit(32) + 4
              = sumDigit(3) + 2 + 4      [sumDigit(32) = sumDigit(3) + 2]
              = 3 + 2 + 4                [sumDigit(3) = 3]
              = 9
```

Note:
It is easy to separate the last digit of an integer using the modulus operator (324 % 10 = 4). Also, you can also use the division operator to separate 32 from 324 (324/10 = 32).

Problem 4: Printing a string backward

In this problem, given a string you need to print it in reverse or backward. That is, if we pass this method "dog" then it should print "god". Printing a string backward can be done recursively. This is similar to what we did in the lecture where we printed all the numbers from 1 to n. To do it recursively, think of the following specification:

```
if word contains any characters (i.e., is not the empty string)
    print the last character in word
    print word' backwards, where word' is word without its last character
```

Problem 5: Palindromes

A palindrome is a string that is the same forward and backward, case insensitive. In our project 01 you saw a program that uses a loop to determine whether a string is a palindrome or not. However, it is also easy to define a palindrome recursively as follows:

- A string containing fewer than 2 letters is always a palindrome.
- A string containing 2 or more letters is a palindrome if
  - its first and last letters are the same, and
  - 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 s
- 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 the index *j*.

So if word is "happy" then word.length is 5, word.charAt(1) is 'a', and word.substring(2,4) is "pp".
For the following two problems you can use the copyOfRange() method from the Arrays class to select a subset of elements in an array.

```
Arrays.copyOfRange(array, 1, 6);
```

The statement above will return a copy of the array which has elements from index 1 to index (6-1) or 5. Please review its documentation to understand what it does. You can use this method or any other methods from the library to get a smaller size array.

Problem 6: Sum of positive elements in an array
Given an array of integers, find the sum of all positive numbers in the array. If we have an array [5, 1, -9, 3, -8] then our recursive method needs to return 9.

Problem 7: Maximum in an array
Given an array of integers you need to find and return the max value. The array will always have at least one item. If we have an array [5, 1, 9, 3, 8] then our recursive method needs to return 9. This can be solved in a recursive way. The idea is to separate one number to reduce the array to a smaller one and find max in the smaller array. Then we can compare separated number with the max in the smaller array to find the max in the actual array.

- Max in [3, 1, 9] is compare 3 and Max in [1, 9]
    - Max in [1, 9] is compare 1 and Max in [9]
        - Max in [1] is 1. As it has only one element.
    - Now, compare 1 and 9. Max is 9
- Now, compare 3 and 9. Max is 9.

JUnit Tester:
Write a JUnit tester called RecusionTester for the Recursion class to test all the methods you implemented. When you test you need to test methods thoroughly including edge cases. You can call static methods using Recursion.factorial(3) or Recursion.power(3,2). You don't need to create an object to call static methods. You don't need to write assert statements for the methods for which return type is void. Just call the method and check console output for checking correctness.

Important Note:
Make sure the file name is correct and code is well commented (Javadoc). No late submission. You will get zero for late submission. You don't need to submit Javadoc.
Submission:
Submit Recursion.java and RecusionTester.java on Canvas.

Grading:

Task	**Points**
Factorial	12
Power	13
Sum of digits of an integer	13
Printing a string backward	13
Palindromes	13
Sum of positive elements in an array	13
Maximum in an array	13
JUnit Tester	10
Total	**100**