Practical Exercise 2 – Recursion

Overall Objective

To write programs that use recursive implementation.

Background

You will need to know:

1. basic Java programming knowledge
2. methods
3. loop concepts
4. **recursion**

Description

Part 1: Discussion

1. What is a recursive method?
   Describe the characteristics of recursive methods. [Refer to slide 29]

2. What is a cause of a stack-overflow exception? [refer to slide 14-25]

3. Which of the following statements are *true*?
   a. Any non-recursive method can be converted into a recursive method.

   b. Non-recursive methods take more resources to execute than recursive methods.

   c. Recursive methods without selection statement lead to infinite recursion.

   d. Recursive definition does not guarantee that a recursive code is the best way to solve a problem.

4. Consider ***ComputeFactorial.java*** program, how many times is the *factorial* method invoked for `factorial(5)`? What is the base case? [refer to slide 5-25]

5. Consider ***ComputeFibonacci.java*** program, how many times is the *fib* method invoked for `fib(3)`? What are the base cases? [refer to slide 27&28]

6. Consider ***RecursivePalindromeUsingSubstring.java*** program, how many times is the *isPalindrome* method invoked for `isPalindrome(abdxcxdba)`? What are the base cases? [refer to slide 31]

7. Consider ***TowersOfHanoi.java*** program, how many times is the *moveDisks* method invoked for `moveDisks(5, 'A', 'B', 'C')`? [refer to slide 41, formula 2^n - 1 where n denotes number of disks]

8. Show the output of the following programs. Also, identify base cases and recursive calls.

i.
```
public class Test {
      public static void main(String[] args) {
            System.out.println("GCD is " + xMethod(48, 18));
      }

      public static int xMethod(int n1, int n2) {
            int remainder;

            remainder = n1 % n2;
            if (remainder == 0)
                  return n2;
            else
                  return xMethod(n2, remainder);
      }
}
```

ii.
```
public class Test {
      public static void main(String[] args) {
            System.out.println("Exponent is " + xMethod(3, 4));
      }

      public static long xMethod(int base, int exponent) {
            if (exponent == 0)
                  return 1;
            else
                  return base * xMethod(base, exponent - 1);
      }
}
```

9. What is wrong in the following recursive solution?

i.
```
public class Test {
      public static void main(String[] args) {
            System.out.println("Factorial = " + factorial(4.0));
      }

      public static long factorial(double n) {
            if (n == 0)
                  return 1;
            else
                  return n * factorial(n - 1);
      }
}
```

ii.
```
public class Test {
      public static void main(String[] args) {
            System.out.println("Exponent is " + power(3, 4));
      }

      public static long power (int base, int exponent) {
            return base * power(base, exponent - 1);
      }
}
```

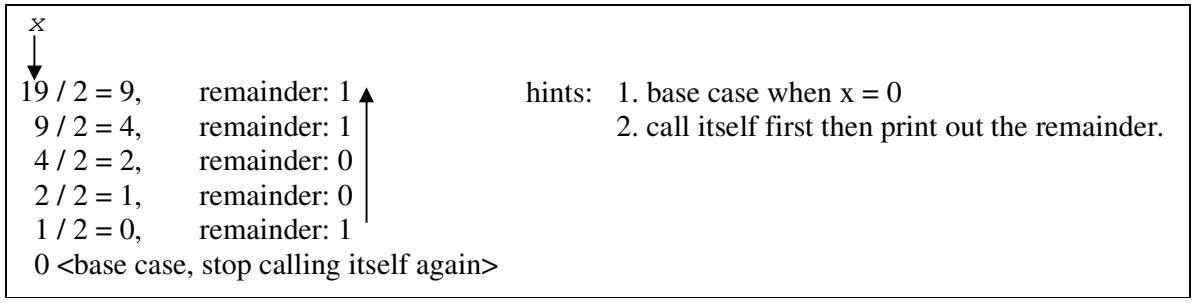
10. Write a recursive definition for printing numbers from ***n*** to **0**.

11. Write a recursive definition for printing numbers from **0** to ***n***.

12. Write a recursive mathematical definition for computing the *sum* of the *squares* of the first ***n*** positive integers.

13. Write a recursive mathematical definition for computing the power x^n for a positive integer ***n*** and a real number ***x***.

14. Write a recursive definition, **`public static void binaryPrint(int x)`** to print out the non-negative integer ***x*** as a binary number. For example, if the value of ***x*** were **`19`** then the binary output should be **`10011`**.



Part 2: Programming Exercise

Run and test the above recursive definitions you have created for Question 10-14. You are required to write main program according to the tasks given to test the recursive methods.

Note that a recursive solution must have two parts: its basis (base case), and its recursive part (general case and convergence). Try to find out these two essential ingredients to recursion before you start to construct your algorithm and program.