PRACTICAL ASSESSMENT – RECURSION (SET 2)

Question 1: Count occurrences of a specified character in string [5 marks]

Write a recursive method that finds the number of occurrences of a specified letter in a string using the following header:

public static int count(String str, char a)

For example: count("programming", 'm') returns 2.

Write a test program that prompts the user to enter string and a character, and display the number of occurrences for the character in the string.

[Answer]

```
import java.util.Scanner;

public class Question1 {

     public static void main(String[] args) {

          Scanner input = new Scanner(System.in);
         System.out.print("Enter a word: ");

           String str =  input.nextLine();

           System.out.print("Enter a character to count in
the word: ");

          // char a =  input.nextChar();
           char a = input.next().charAt(0);

           Question1 word = new Question1();
           System.out.println("Occurrence of char '" + a +
"' in word '" + str + "' is : "+ word.count(str, a));

     }

     public static int count(String str, char a) {
          if (str.length() == 0)
              return 0;
            else if (str.charAt(0) == a)
              return 1 + count(str.substring(1), a);
            else
              return count(str.substring(1), a);

     }

}
```

Question 2: Print characters [5marks]

Write and test a recursive method that displays any given character for specified number of times (n) until it reaches n=1.

For example: User input: [* , 5] , the output should be:

```
* * * * *
* * * *
* * *
* *
*
```

[Answer]

```
import java.util.Scanner;

public class Question2 {

      public static void main(String[] args) {
            Scanner input = new Scanner(System.in);
           System.out.print("Enter a char: ");
            char a = input.next().charAt(0);

System.out.print("Specify number of times it should print: ");
             int n = input.nextInt();

             printPatternRecur(a, n);

      }

      static void printPatternRecur(char a, int n) {
         // base condition
         if (n < 1)
             return;

         // print the stars of the n-th row
         printNTimes(a, n);

         // move to next line
         System.out.println ();

         // print stars of the remaining rows recursively
         printPatternRecur(a,n - 1);

    }
    static void printNTimes(char a, int n) {
         // base condition
         if (n < 1)
             return;

         // print the remnaining stars of the n-th row recursively
         System.out.print(a);
         printNTimes(a, n - 1);
    }
}
```