Question 1

Consider the following TowerOfHanoi program:

```java
public class TowerOfHanoi {
    public static void main(String[] args) {
        moveDisks(3, 'A', 'B', 'C');
    }

    /** The method for finding the solution to move n disks
        from fromTower to toTower with auxTower */
    public static void moveDisks(int n, char fromTower, char toTower, char auxTower) {
        if (n == 1) // Stopping condition
            System.out.println("Move disk " + n + " from " + fromTower + " to " + toTower);
        else {
            moveDisks(n - 1, fromTower, auxTower, toTower);
            System.out.println("Move disk " + n + " from " + fromTower + " to " + toTower);
            moveDisks(n - 1, auxTower, toTower, fromTower);
        }
    }
}
```

1. How many times is the function moveDisks invoked?

2. Sketch what calls are on the run-time stack right after the message “Move disk 1 from C to A” is printed out.

Question 2

Implement each of the following functions using recursion in Java.

1. static int sum_of_squares(int n): Return the summation of squares \( \sum_{k=1}^{n} (k^2) \).
2. static int gcd(int x, int y): Return the greatest common divisor of x and y.
3. static int sumEven(int[] a): Return the sum of the even integers in array a.
4. static boolean sorted(int[] a): Test if array a is sorted in ascending order.
5. static int[] copy(int[] a): Return a copy of array a.
6. public static int count(String str, char a): Return the number of occurrences of character a in string str.
7. static long bin_str_to_int(String binStr): Convert an unsigned binary string to a decimal integer. For example, for binStr = 11010, the returned integer is 26.