JAVA Lab
Lab Experiment number 03

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Write a menu driven program to implement recursive Functions for following tasks:

a) To find GCD and LCM
b) To print n Fibonacci numbers
c) To find reverse of number
d) To solve 1 +2+3+4+.....+(n- l)+n

Theory:

Recursion:

Recursion is the technique of making a function call itself. This technique provides a way to break complicated problems down into simple problems which are easier to solve.

1. Recursion is a process in which the problem is specified in terms of itself.
2. The function should be called itself to implement recursion.
3. The function which calls itself is known as a recursive function.
4. A condition must be specified to stop recursion; otherwise it will lead to an infinite process.
5. In case of recursion, all partial solutions are combined to obtain the final solution.

Iteration and Recursion, comparison:

Time Complexity: Finding the Time complexity of Recursion is more difficult than that of Iteration.

> **Recursion**: Time complexity of recursion can be found by finding the value of the nth recursive call in terms of the previous calls. Thus, finding the destination case in terms of the base case, and solving in terms of the base case gives us an idea of the time complexity of recursive equations.
>
> **Iteration**: Time complexity of iteration can be found by finding the number of cycles being repeated inside the loop.

Usage: Usage of either of these techniques is a trade-off between time complexity and size of code. If time complexity is the point of focus, and the number of recursive calls would be large, it is better to use iteration. However, if time complexity is not an issue and shortness of code is, recursion would be the way to go.

> **Recursion**: Recursion involves calling the same function again, and hence, has a very small length of code. However, as we saw in the analysis, the time complexity of recursion can get to be exponential when there are a considerable number of recursive calls. Hence, usage of recursion is advantageous in shorter code, but higher time complexity.
>
> **Iteration**: Iteration is repetition of a block of code. This involves a larger size of code, but the time complexity is generally lesser than it is for recursion.

Overhead: Recursion has a large amount of Overhead as compared to Iteration.

Recursion: Recursion has the overhead of repeated function calls, that is due to repetitive calling of the same function, the time complexity of the code increases manifold.

Iteration: Iteration does not involve any such overhead.

Infinite Repetition: Infinite Repetition in recursion can lead to CPU crash but in iteration, it will stop when memory is exhausted.

Recursion: In Recursion, Infinite recursive calls may occur due to some mistake in specifying the base condition, which on never becoming false, keeps calling the function, which may lead to system CPU crash.

Iteration: Infinite iteration due to mistake in iterator assignment or increment, or in the terminating condition, will lead to infinite loops, which may or may not lead to system errors, but will surely stop program execution any further.

Example of Recursion:

The simple function count demonstrates the use of recursion

```
function Count (integer N)
        if (N <= 0) return "Must be a Positive Integer";
        if (N > 9) return "Counting Completed";
        else return Count (N+1);
end function
```

The function Count() above uses recursion to count from any number between 1 and 9, to the number 10. For example, Count(1) would return 2,3,4,5,6,7,8,9,10. Count(7) would return 8,9,10. The result could be used as a roundabout way to subtract the number from 10.

Advantages and disadvantages of Recursion:

Advantages

1. The main benefit of a recursive approach to algorithm design is that it allows programmers to take advantage of the repetitive structure present in many problems.
2. Complex case analysis and nested loops can be avoided.
3. Recursion can lead to more readable and efficient algorithm descriptions.
4. Recursion is also a useful way for defining objects that have a repeated similar structural form.

Disadvantages

1. Slowing down execution time and storing on the run-time stack more things than required in a non recursive approach are major limitations of recursion.
2. If recursion is too deep, then there is a danger of running out of space on the stack and ultimately the program crashes.
3. Even if some recursive function repeats the computations for some parameters, the run time can be prohibitively long even for very simple cases.

Program:

```
//code

import java.util.*;

class Recursion {
```

```
static int sum = 0, rem;

// method to calculate GCD
static int gcd(int n1, int n2) {

    if (n2 != 0) {
        return gcd(n2, n1%n2);
    } else {
        return n1;
    }

}

// Fibonacci
static int fibo(int n) {

    if (n==0 || n==1) {
        return n;
    } else {
        return (fibo(n-1) + fibo(n-2));
    }

}

// Reverse
static int reverse(int n) {

    if (n != 0) {
        rem = n % 10;
        sum = sum*10 + rem;
        reverse(n/10);
    } else {
        return sum;
    }
    return sum;

}

// Ramanujan series
```

```
static int seriesSum (int n) {

    if (n != 0) {
        sum += n;
        return seriesSum(n-1);
    }
    return sum;

}

public static void main(String args[]) {

    Scanner sc = new Scanner(System.in);
    int choice;

    while (true) {
        // choices
        System.out.println("**********************");
        System.out.println("*1 GCD and LCM");
        System.out.println("*2 Fibonacci");
        System.out.println("*3 Reverse");
        System.out.println("*4 Ramanujan series");
        System.out.println("*5 EXIT");
        System.out.print("Enter Your choice :  ");
        choice = sc.nextInt();

        switch (choice) {

            case 1:     // GCD and LCM
                System.out.print("Enter First number  :  ");
                int num1 = sc.nextInt();
                System.out.print("Enter Second number :  ");
                int num2 = sc.nextInt();

                // get GCD
                int gcd = gcd(num1, num2);

                // get LCM
                int lcm = (num1 * num2) / gcd;
```

```
    // printing
    System.out.println("GCD of "+num1+" and "+num2+" is :  "+gcd);
    System.out.println("LCM of "+num1+" and "+num2+" is :  "+lcm);

    sum=0;
    break;

case 2:     // Fibonacci
    // input
    int num;
    System.out.print("Enter number of elements required in Fibonacci series :  ");
    num = sc.nextInt();

    // print series
    for (int i=0 ; i<num ; i++) {
       System.out.print(fibo(i)+"  ");
    }
    System.out.println();

    sum=0;
    break;

case 3:     // Reverse
    // input
    int reverse;
    System.out.print("Enter a number to Reverse :  ");
    num = sc.nextInt();
    reverse = reverse(num);

    System.out.println("Reverse of "+num+" is :  "+reverse);

    sum=0;
    break;

case 4:     //Ramanujan series
    // input
    int numRam=0;
    System.out.print("Enter value of n in Ramanujan summation series :  ");
    numRam = sc.nextInt();
```

```
                // display
                System.out.println("Sum of "+numRam+" terms in Ramanujan series is :
"+seriesSum(numRam));
                sum=0;
                break;

            case 5:     // Exit
                System.out.println("*** E X I T I N G ***");
                System.exit(1);

            default:
                System.out.println("INVALID INPUT");

        }
    }
  }
}
```

Output

```
E:\Aamir\Sem-3\LabWork - Assignments\OOPM\Lab Assignment 3\Code>javac Recursion.java

E:\Aamir\Sem-3\LabWork - Assignments\OOPM\Lab Assignment 3\Code>java Recursion
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  1
Enter First number  :  15
Enter Second number :  20
GCD of 15 and 20 is :  5
LCM of 15 and 20 is :  60
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  1
Enter First number  :  3
Enter Second number :  4
GCD of 3 and 4 is :  1
LCM of 3 and 4 is :  12
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  2
Enter number of elements required in Fibonacci series :  5
0  1  1  2  3
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  2
Enter number of elements required in Fibonacci series :  10
0  1  1  2  3  5  8  13  21  34
********************
```

```
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  3
Enter a number to Reverse :  12345
Reverse of 12345 is :  54321
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  3
Enter a number to Reverse :  52384
Reverse of 52384 is :  48325
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  4
Enter value of n in Ramanujan summation series :  5
Sum of 5 terms in Ramanujan series is :  15
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  4
Enter value of n in Ramanujan summation series :  8
Sum of 8 terms in Ramanujan series is :  36
********************
*1 GCD and LCM
*2 Fibonacci
*3 Reverse
*4 Ramanujan series
*5 EXIT
Enter Your choice :  5
*** E X I T I N G ***
```