Aror University of Art, Architecture, Design and Heritage
SUKKUR, Sindh
Department of Multimedia and Gaming
Course: Data Structures CSC-221 (Practical)
Instructor: Engr. Fatima Jaffar

LAB 13

Objective: Working with Recursion

Introduction:

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.

Recursion may be a bit difficult to understand. The best way to figure out how it works is to experiment with it.

Recursion Example:

Adding two numbers together is easy to do, but adding a range of numbers is more complicated. In the following example, recursion is used to add a range of numbers together by breaking it down into the simple task of adding two numbers:

```
public class Main {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
  }
  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
}
```

Example Explained:

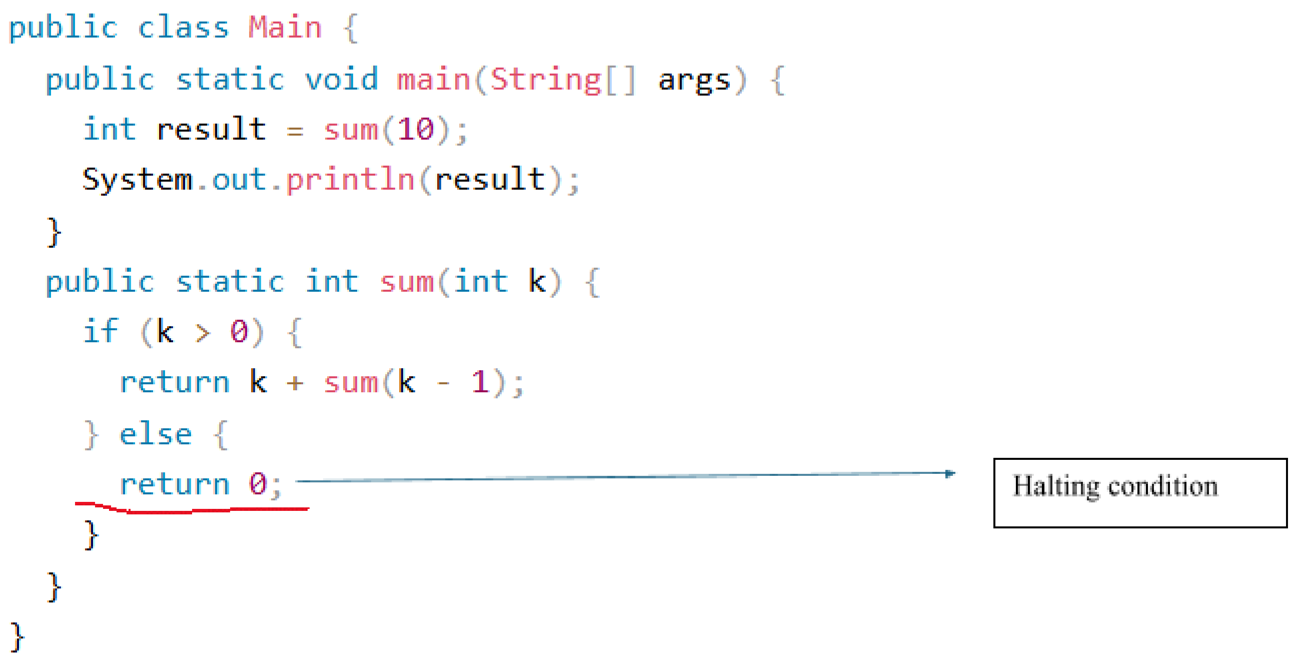
When the sum() function is called, it adds parameter k to the sum of all numbers smaller than k and returns the result. When k becomes 0, the function just returns 0. When running, the program follows these steps:

10 + sum(9)
10 + (9 + sum(8))
10 + (9 + (8 + sum(7)))
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0

Halting Condition:

Just as loops can run into the problem of infinite looping, recursive functions can run into the problem of infinite recursion. Infinite recursion is when the function never stops calling itself.

Every recursive function should have a halting condition, which is the condition where the function stops calling itself. In the previous example, the halting condition is when the parameter k becomes 0.



```java
public class Main {
  public static void main(String[] args) {
    int result = sum(10);
    System.out.println(result);
  }
  public static int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
}
```

Lab Tasks

1. Implement a recursive function to calculate the factorial of a given non-negative integer.
2. Write a recursive function to find the nth Fibonacci number.
3. Write a recursive function to sum the digits of a non-negative integer.
4. Write a recursive function to calculate the power of a number raised to a non-negative integer exponent.
5. Develop a recursive function to find the maximum element in an integer array.
6. Develop a recursive function to find the length of a string.
7. Implement binary search using recursion.