Lab 25

Self Practice Solution: Write three programs based on Java recursion

1. Calculating Factorial using Recursion:

public class FactorialCalculator {

// Recursive method to calculate factorial

static int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

public static void main(String[] args) {

int num = 5;

int result = factorial(num);

System.out.println("Factorial of " + num + " is " + result); // Output: Factorial of 5 is 120

}

}

In this program, the `factorial` method is defined recursively to calculate the factorial of a number. The base case (when `n` is 0 or 1) terminates the recursion, and the method gradually computes the factorial by calling itself with a smaller value of `n`.

2. Calculating Fibonacci Sequence using Recursion:

public class FibonacciCalculator {

// Recursive method to calculate Fibonacci number

static int fibonacci(int n) {

if (n <= 1) {

return n;

} else {

return fibonacci(n - 1) + fibonacci(n - 2);

}

}

public static void main(String[] args) {

int n = 7;

System.out.print("Fibonacci Sequence up to " + n + ": ");

for (int i = 0; i < n; i++) {

System.out.print(fibonacci(i) + " ");

}

// Output: Fibonacci Sequence up to 7: 0 1 1 2 3 5 8

}

}

In this program, the `fibonacci` method is defined recursively to calculate Fibonacci numbers. The base case (when `n` is 0 or 1) returns the corresponding Fibonacci number, and the method recursively computes subsequent Fibonacci numbers by calling itself with smaller values of `n`.

3. Recursive Binary Search:

public class BinarySearch {

// Recursive binary search method

static int binarySearch(int[] arr, int target, int left, int right) {

if (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) {

return mid;

} else if (arr[mid] < target) {

return binarySearch(arr, target, mid + 1, right);

} else {

return binarySearch(arr, target, left, mid - 1);

}

}

return -1; // Element not found

}

public static void main(String[] args) {

int[] arr = {1, 3, 5, 7, 9, 11, 13};

int target = 7;

int result = binarySearch(arr, target, 0, arr.length - 1);

if (result != -1) {

System.out.println("Element found at index " + result); // Output: Element found at index 3

} else {

System.out.println("Element not found.");

}

}

}

This program demonstrates a recursive binary search algorithm to find a target element in a sorted array. The method `binarySearch` recursively divides the search range in half and narrows down the search until the target element is found or the search range is empty.