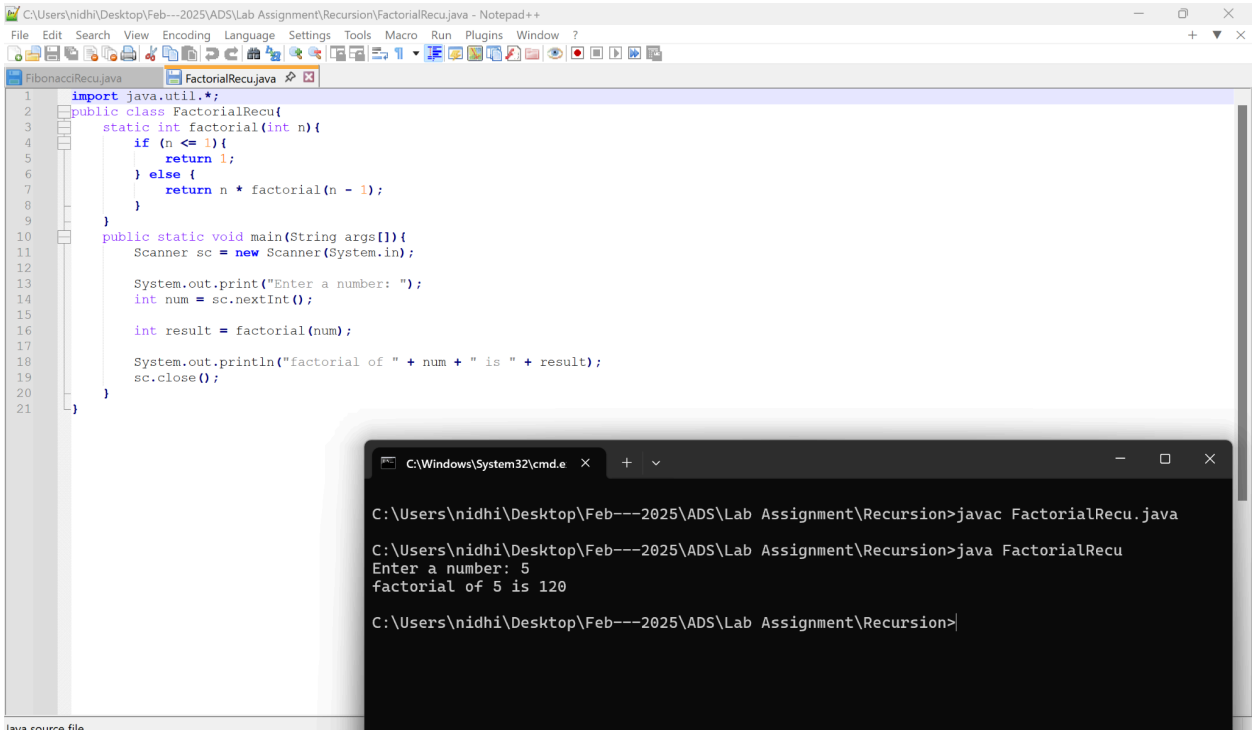
Recursion Assignment

Que 1 : Factorial



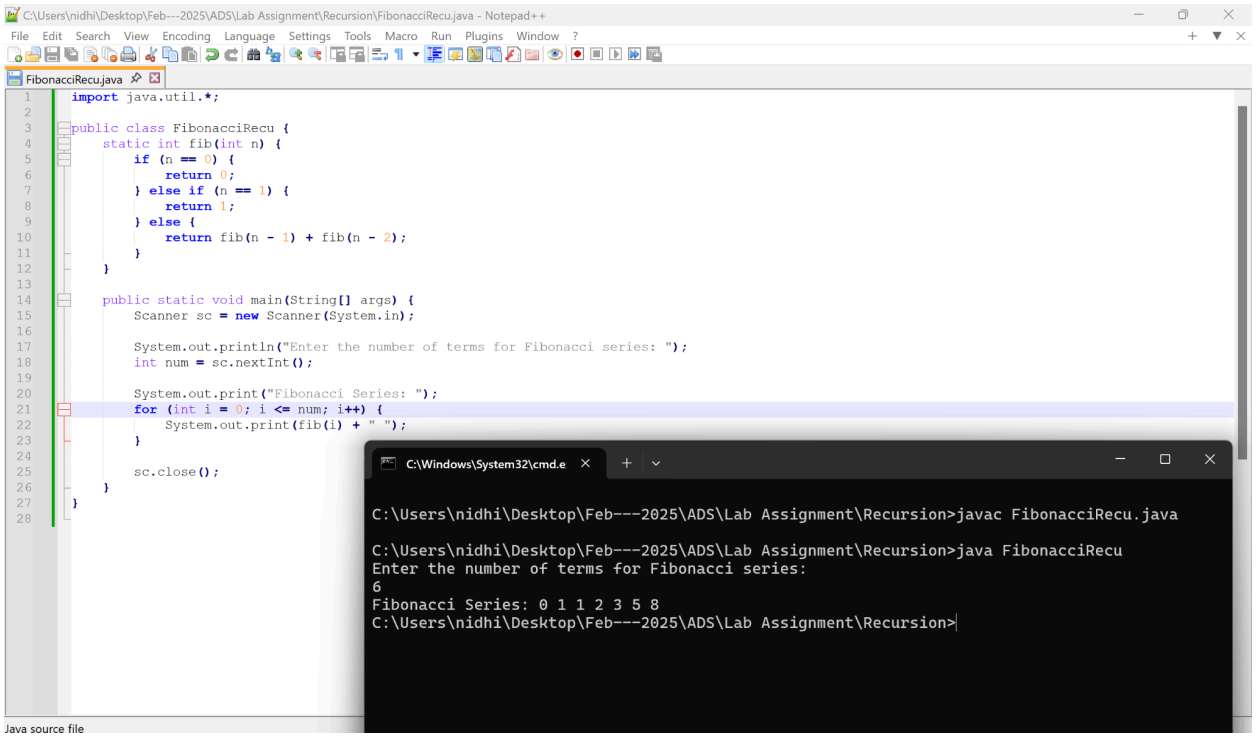
Explanation :

The recursion in the above code will run in the following manner,

Example of Factorial of 5
factorial(5) = 5 * factorial(4)
factorial(4) = 4 * factorial(3)
factorial(3) = 3 * factorial(2)
factorial(2) = 2 * factorial(1)
factorial(1) = 1 (Base case reached)

Then the function returns and evaluates as:
factorial(2) = 2 * 1 = 2
factorial(3) = 3 * 2 = 6
factorial(4) = 4 * 6 = 24
factorial(5) = 5 * 24 = 120

Que 2 : Fibonacci

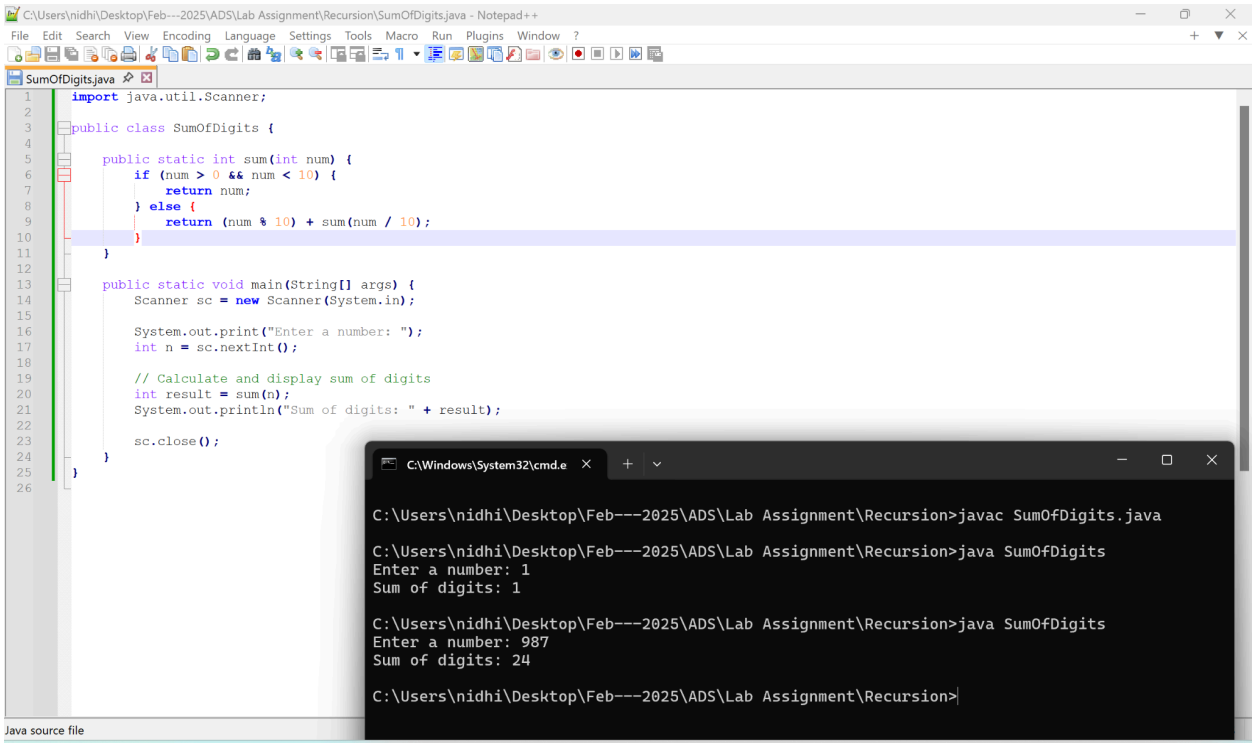


Explanation :

The recursion in the above code will run in the following manner,
fib(6) = fib(5) + fib(4)
fib(5) = fib(4) + fib(3)
fib(4) = fib(3) + fib(2)
fib(3) = fib(2) + fib(1)
fib(2) = fib(1) + fib(0)
fib(1) = 1 (Base case)
fib(0) = 0 (Base case)

fib(0) = 0
fib(1) = 1
fib(2) = 1 + 0 = 1
fib(3) = 1 + 1 = 2
fib(4) = 2 + 1 = 3
fib(5) = 3 + 2 = 5
fib(6) = 5 + 3 = 8

Que 3 : Sum of Digit

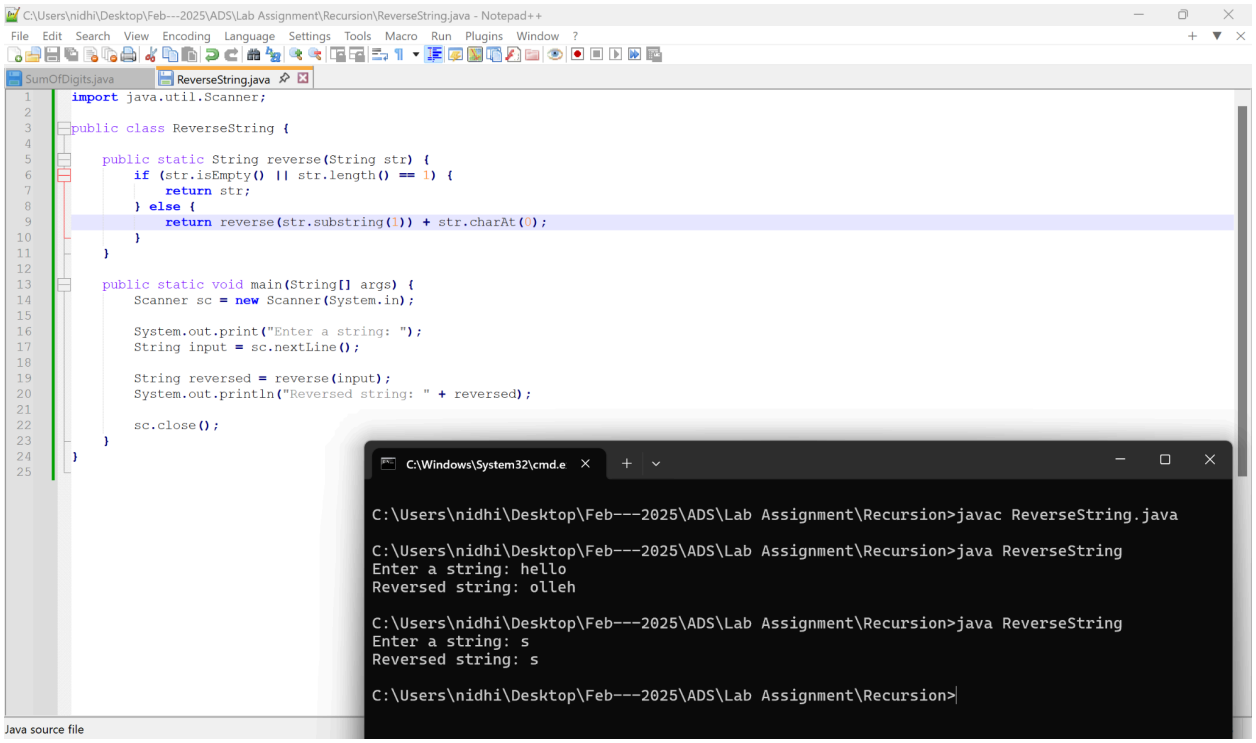


If condition will check for the single digit number and return the number itself.
Else will recursively check till single number

sum(987) = sum(98) + 7
sum(98) = sum(9) + 8
sum(9) = 9 (Base case)

sum(9) = 9
sum(98) = 9 + 8 = 17
sum(987) = 17 + 7 = 24

Que 4 : Reverse The String

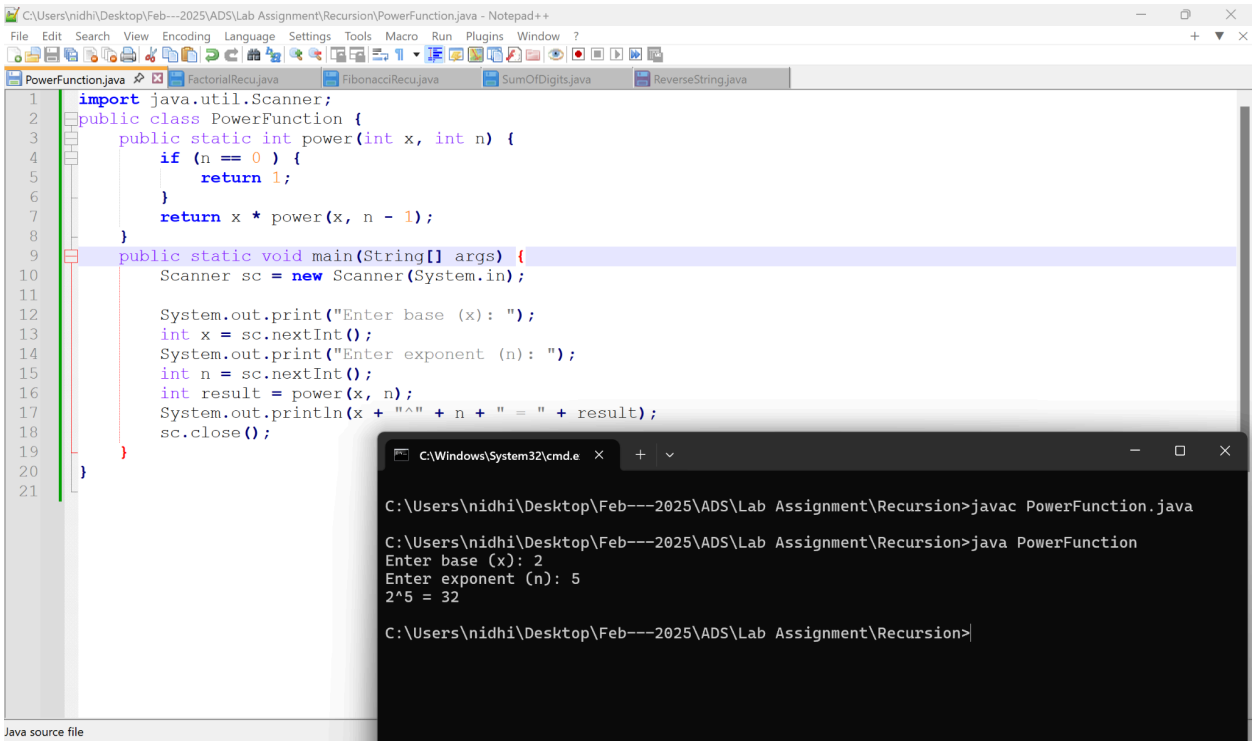


Base Case: If the string is empty or a single character is there it returns the string itself.
Recursive Case:
Take the substring from index 1 onward (str.substring(1))
Call reverse() on this substring.
Append the first character (str.charAt(0)) to the reversed substring.
The recursion continues until only one character remains.

reverse("hello") = reverse("ello") + 'h'
reverse("ello") = reverse("llo") + 'e'
reverse("llo") = reverse("lo") + 'l'
reverse("lo") = reverse("o") + 'l'
reverse("o") = reverse("") + 'o' (Base case reached)

Returning from recursion:
reverse("o") = "o"
reverse("lo") = "o" + "l" = "ol"
reverse("llo") = "ol" + "l" = "oll"
reverse("ello") = "oll" + "e" = "olle"
reverse("hello") = "olle" + "h" = "olleh"

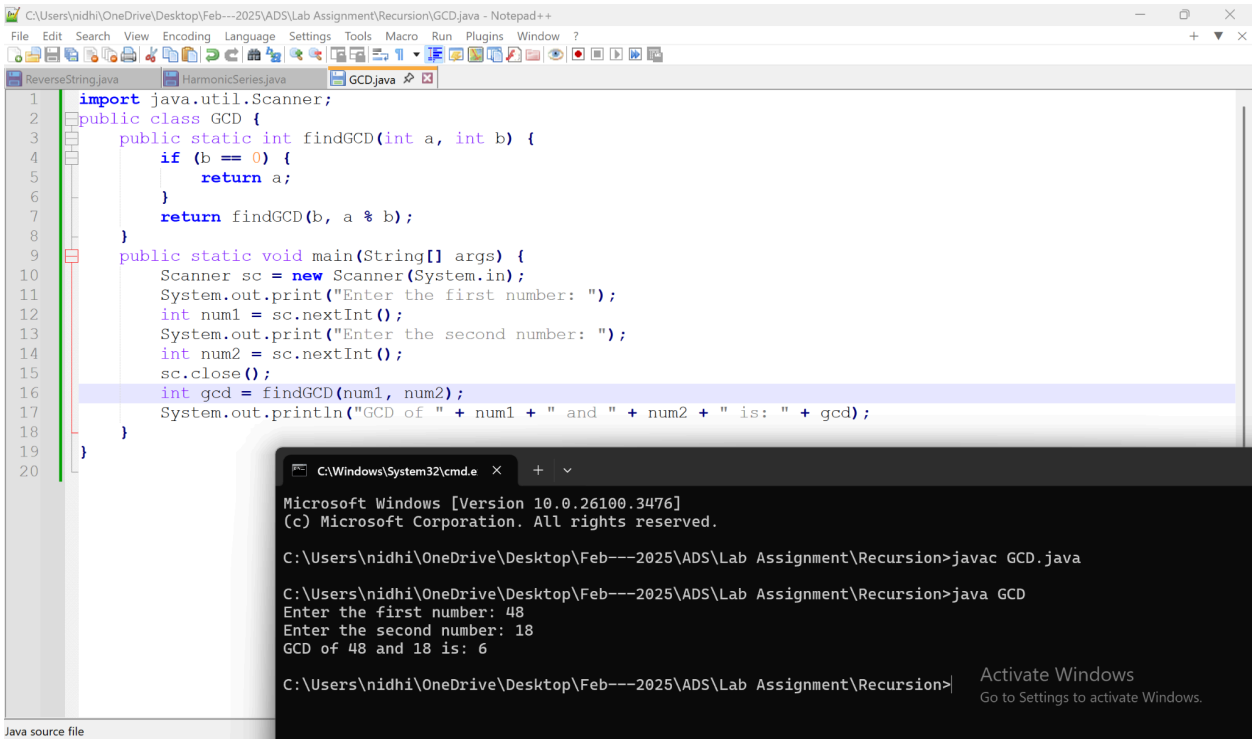
Que 5 : Power Function



power(2, 5) = 2 * power(2, 4)
power(2, 4) = 2 * power(2, 3)
power(2, 3) = 2 * power(2, 2)
power(2, 2) = 2 * power(2, 1)
power(2, 1) = 2 * power(2, 0)
power(2, 0) = 1 (Base case)

Returning from recursion:
power(2, 1) = 2 * 1 = 2
power(2, 2) = 2 * 2 = 4
power(2, 3) = 2 * 4 = 8
power(2, 4) = 2 * 8 = 16
power(2, 5) = 2 * 16 = 32

Que 6 : GCD using Recursive



Explanation:

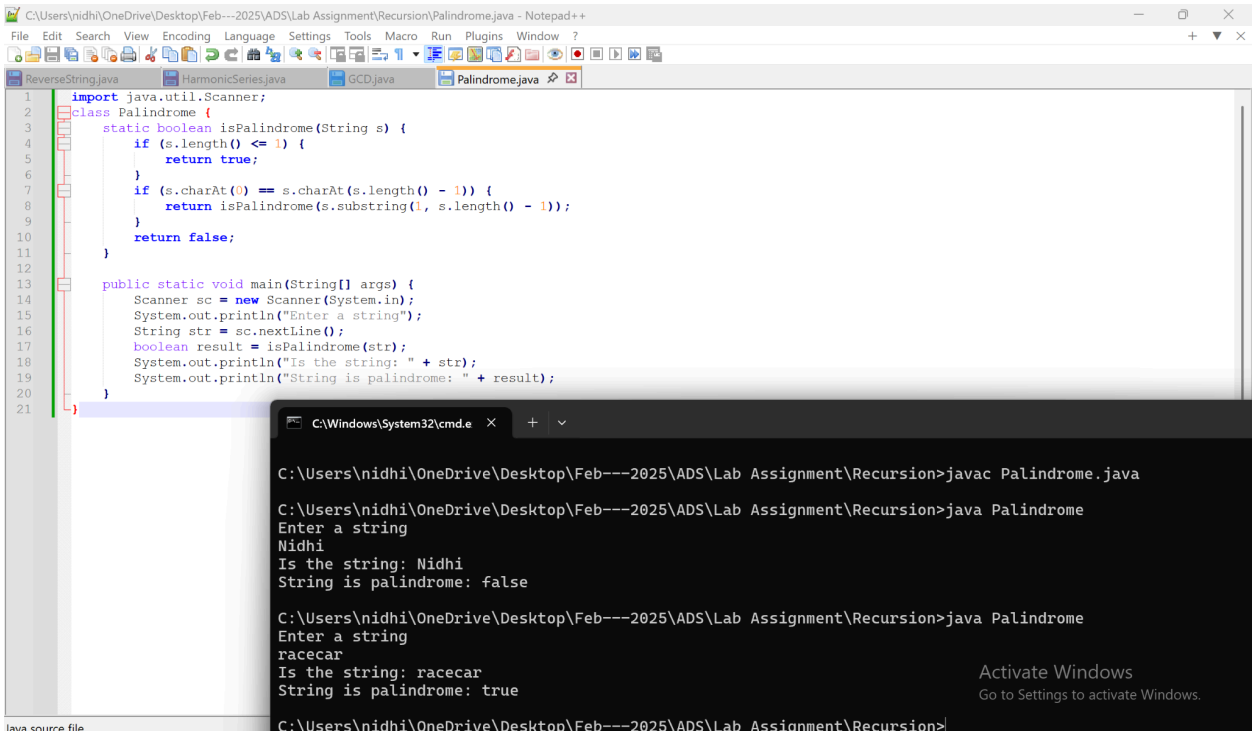
1. Base Case: If `b == 0`, return `a` (GCD found)
2. Recursive Case: Call `findGCD(b, a % b)`, reducing the problem size
3. Example Calculation for (48, 18):
   findGCD(48, 18) = findGCD(18, 48 % 18) = findGCD(18, 12)

   findGCD(18, 12) = findGCD(12, 18 % 12) =findGCD(12, 6)

   findGCD(12, 6) = findGCD(6, 12 % 6) = findGCD(6, 0)

   findGCD(6, 0) returns 6 (final answer).

Que 7 : Palindrome String

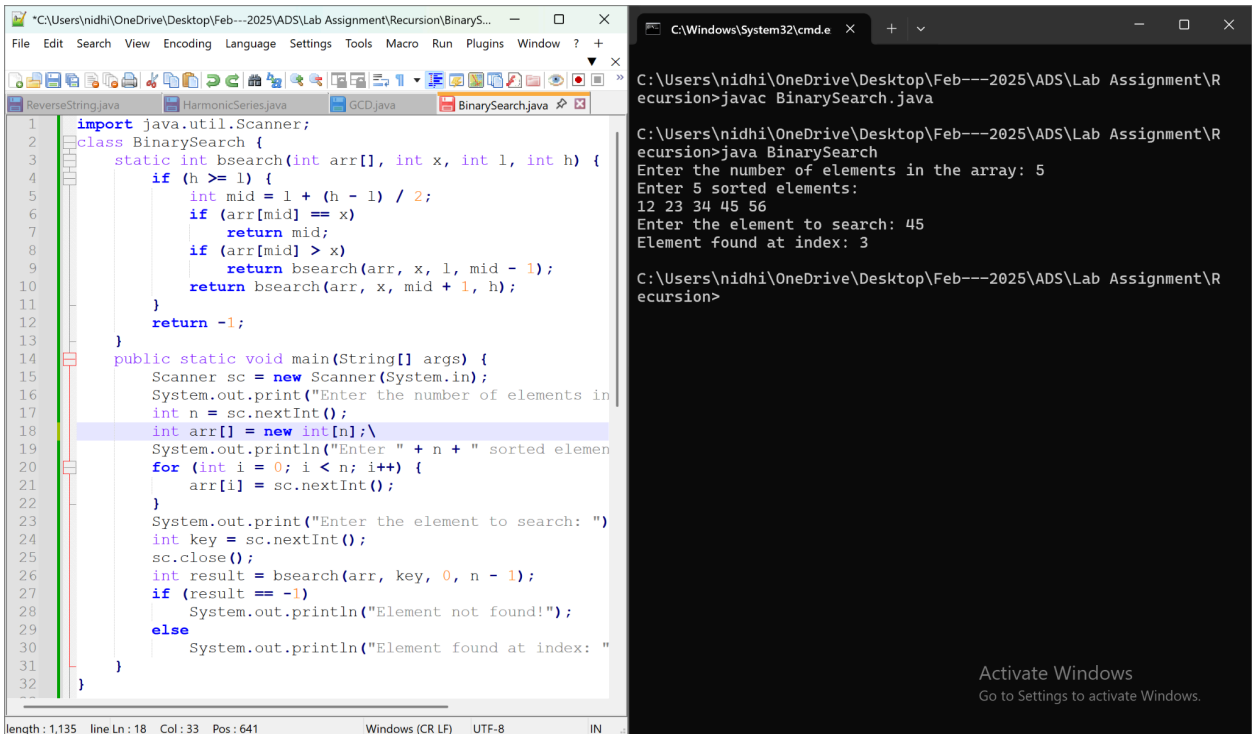


```
import java.util.Scanner;
class Palindrome {
    static boolean isPalindrome(String s) {
        if (s.length() <= 1) {
            return true;
        }
        if (s.charAt(0) == s.charAt(s.length() - 1)) {
            return isPalindrome(s.substring(1, s.length() - 1));
        }
        return false;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter a string");
        String str = sc.nextLine();
        boolean result = isPalindrome(str);
        System.out.println("Is the string: " + str);
        System.out.println("String is palindrome: " + result);
    }
}
```

```
C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac Palindrome.java

C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java Palindrome
Enter a string
Nidhi
Is the string: Nidhi
String is palindrome: false

C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java Palindrome
Enter a string
racecar
Is the string: racecar
String is palindrome: true
```

Function Call	String	First & Last Match?	Recursive Call
`isPalindrome("racecar")`	`"racecar"`	`'r' == 'r'`	`isPalindrome("aceca")`
`isPalindrome("aceca")`	`"aceca"`	`'a' == 'a'`	`isPalindrome("cec")`
`isPalindrome("cec")`	`"cec"`	`'c' == 'c'`	`isPalindrome("e")`
`isPalindrome("e")`	`"e"`	(Base case)	`true`

Que 8 : Binary Search



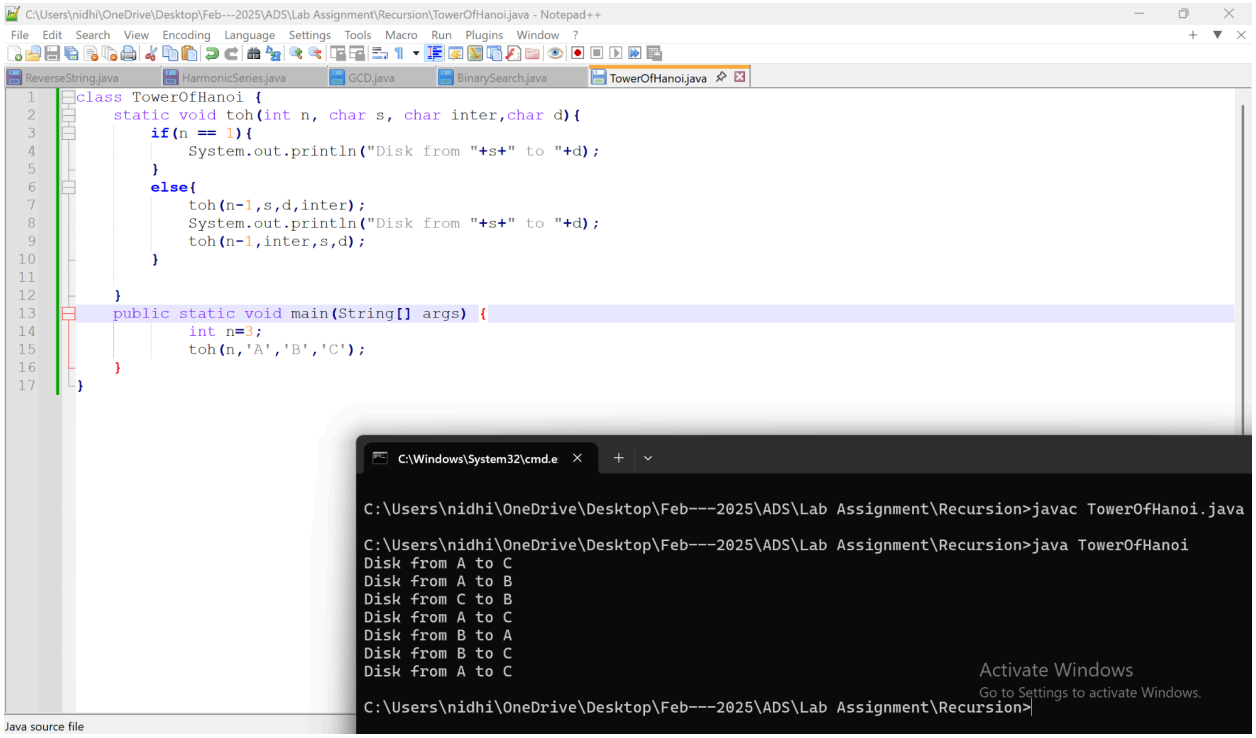
```
import java.util.Scanner;
class BinarySearch {
    static int bsearch(int arr[], int x, int l, int h) {
        if (h >= l) {
            int mid = l + (h - l) / 2;
            if (arr[mid] == x)
                return mid;
            if (arr[mid] > x)
                return bsearch(arr, x, l, mid - 1);
            return bsearch(arr, x, mid + 1, h);
        }
        return -1;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements in
        int n = sc.nextInt();
        int arr[] = new int[n];\
        System.out.println("Enter " + n + " sorted elemen
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }
        System.out.print("Enter the element to search: ")
        int key = sc.nextInt();
        sc.close();
        int result = bsearch(arr, key, 0, n - 1);
        if (result == -1)
            System.out.println("Element not found!");
        else
            System.out.println("Element found at index: "
    }
}
```

```
C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\R
ecursion>javac BinarySearch.java

C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\R
ecursion>java BinarySearch
Enter the number of elements in the array: 5
Enter 5 sorted elements:
12 23 34 45 56
Enter the element to search: 45
Element found at index: 3

C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\R
ecursion>
```

User Input:
Takes n (number of elements).
Takes n sorted elements in the array.
Take the key to search.

Recursive Binary Search:
Finds mid index.
If arr[mid] == key, return index.
If arr[mid] > key, search left half.
Else, search right half.
If l > h, return -1 (not found).

Que 9 : Tower of Hanoi



Step-by-Step Execution for n = 3

We have 3 disks and 3 pegs:

- A → Source
- B → Intermediate
- C → Destination

Recursive Calls Breakdown

1. Move 2 disks from A → B (using C as auxiliary).
2. Move disk 3 from A → C.
3. Move 2 disks from B → C (using A as auxiliary).

Step	Function Call	Action
1	toh(3, A, B, C)	Move 3 disks from A → C using B as auxiliary

2	toh(2, A, C, B)	Move 2 disks from A → B using C
3	toh(1, A, B, C)	Move disk from A → B
4	Print "Disk from A to C"	Move largest disk (disk 3) from A → C
5	toh(1, B, A, C)	Move disk from B → C
6	Print "Disk from A to B"	Move disk 2 from A → B
7	toh(2, C, A, B)	Move 2 disks from C → B using A
8	toh(1, C, B, A)	Move disk from C → A
9	Print "Disk from C to B"	Move largest disk (disk 2) from C → B
10	toh(1, A, C, B)	Move disk from A → B