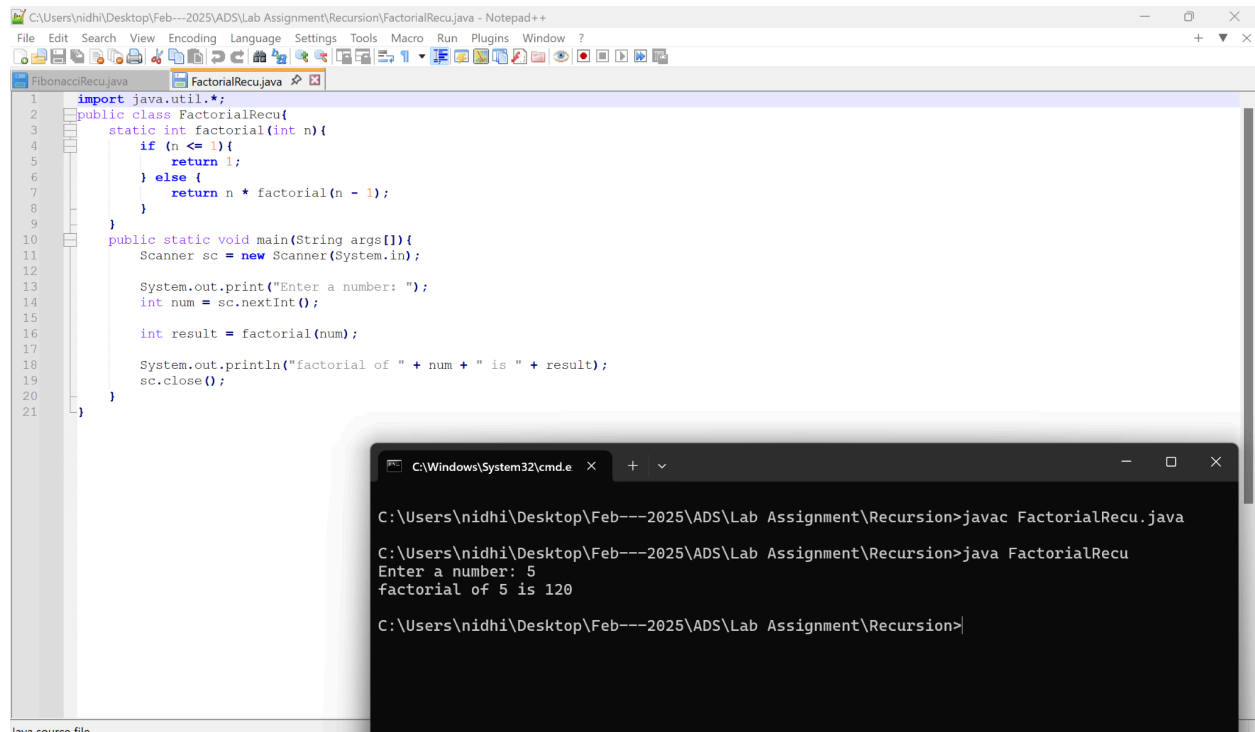


Recursion Assignment

Que 1 : Factorial



```
1 import java.util.*;
2 public class FactorialRecu{
3     static int factorial(int n){
4         if (n <= 1){
5             return 1;
6         } else {
7             return n * factorial(n - 1);
8         }
9     }
10    public static void main(String args[]){
11        Scanner sc = new Scanner(System.in);
12
13        System.out.print("Enter a number: ");
14        int num = sc.nextInt();
15
16        int result = factorial(num);
17
18        System.out.println("factorial of " + num + " is " + result);
19        sc.close();
20    }
21 }
```

```
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac FactorialRecu.java
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java FactorialRecu
Enter a number: 5
factorial of 5 is 120
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>
```

Explanation :

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

Example of Factorial of 5

$\text{factorial}(5) = 5 * \text{factorial}(4)$

$\text{factorial}(4) = 4 * \text{factorial}(3)$

$\text{factorial}(3) = 3 * \text{factorial}(2)$

$\text{factorial}(2) = 2 * \text{factorial}(1)$

$\text{factorial}(1) = 1$ (Base case reached)

Then the function returns and evaluates as:

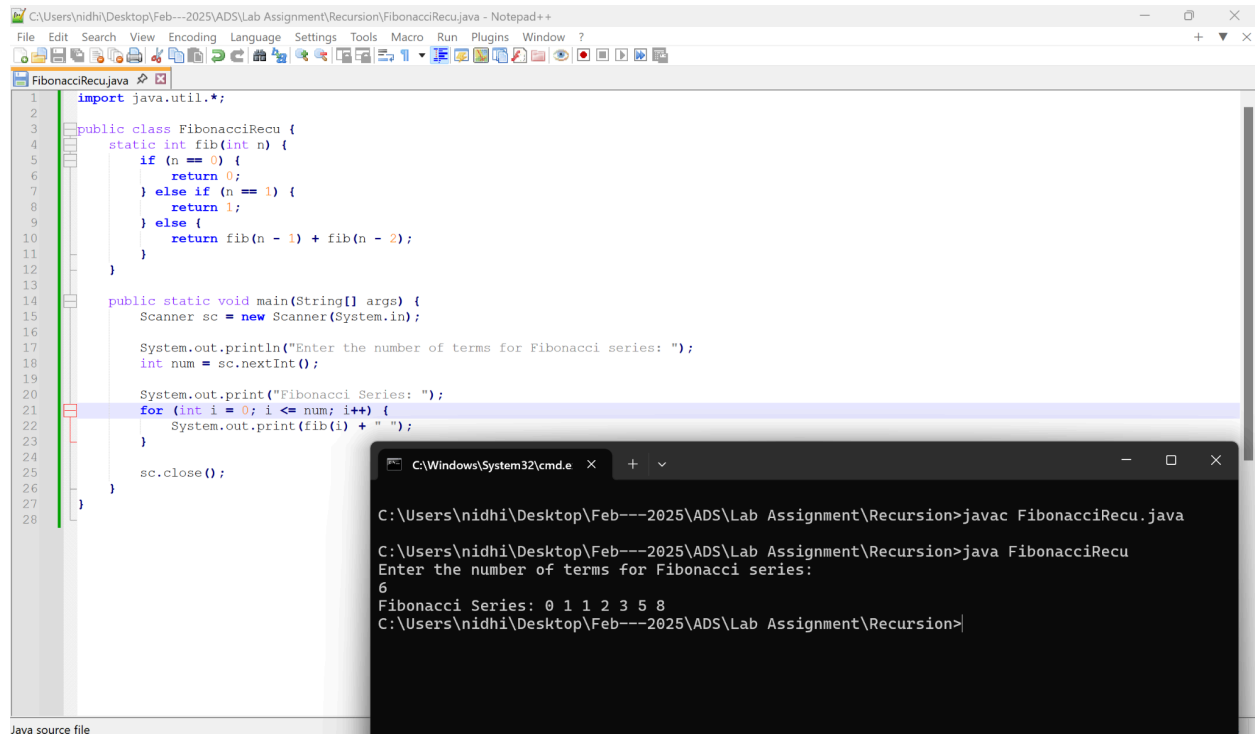
$\text{factorial}(2) = 2 * 1 = 2$

$\text{factorial}(3) = 3 * 2 = 6$

$\text{factorial}(4) = 4 * 6 = 24$

$\text{factorial}(5) = 5 * 24 = 120$

Que 2 : Fibonacci



```
1 import java.util.*;
2
3 public class FibonacciRecu {
4     static int fib(int n) {
5         if (n == 0) {
6             return 0;
7         } else if (n == 1) {
8             return 1;
9         } else {
10            return fib(n - 1) + fib(n - 2);
11        }
12    }
13
14    public static void main(String[] args) {
15        Scanner sc = new Scanner(System.in);
16
17        System.out.println("Enter the number of terms for Fibonacci series: ");
18        int num = sc.nextInt();
19
20        System.out.print("Fibonacci Series: ");
21        for (int i = 0; i <= num; i++) {
22            System.out.print(fib(i) + " ");
23        }
24
25        sc.close();
26    }
27
28 }
```

```
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac FibonacciRecu.java
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java FibonacciRecu
Enter the number of terms for Fibonacci series:
6
Fibonacci Series: 0 1 1 2 3 5 8
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>
```

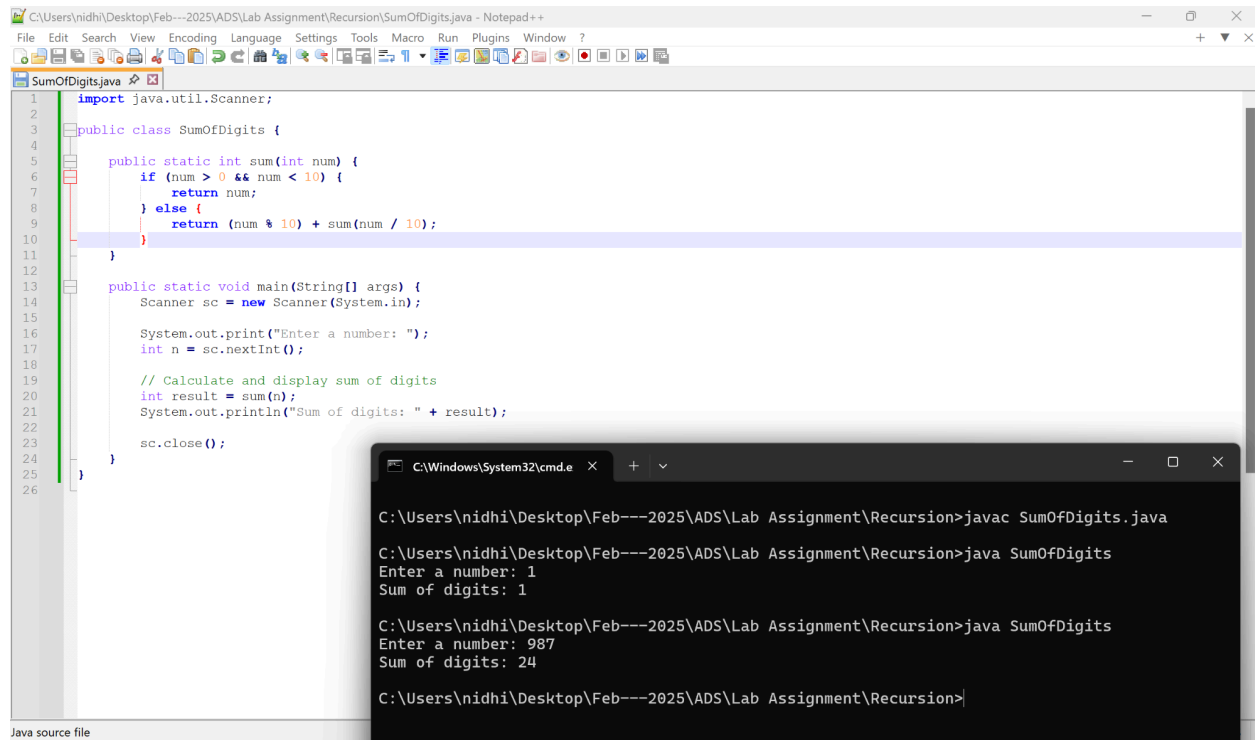
Explanation :

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

$\text{fib}(6) = \text{fib}(5) + \text{fib}(4)$
 $\text{fib}(5) = \text{fib}(4) + \text{fib}(3)$
 $\text{fib}(4) = \text{fib}(3) + \text{fib}(2)$
 $\text{fib}(3) = \text{fib}(2) + \text{fib}(1)$
 $\text{fib}(2) = \text{fib}(1) + \text{fib}(0)$
 $\text{fib}(1) = 1$ (Base case)
 $\text{fib}(0) = 0$ (Base case)

$\text{fib}(0) = 0$
 $\text{fib}(1) = 1$
 $\text{fib}(2) = 1 + 0 = 1$
 $\text{fib}(3) = 1 + 1 = 2$
 $\text{fib}(4) = 2 + 1 = 3$
 $\text{fib}(5) = 3 + 2 = 5$
 $\text{fib}(6) = 5 + 3 = 8$

Que 3 : Sum of Digit



```
1 import java.util.Scanner;
2
3 public class SumOfDigits {
4
5     public static int sum(int num) {
6         if (num > 0 && num < 10) {
7             return num;
8         } else {
9             return (num % 10) + sum(num / 10);
10        }
11    }
12
13    public static void main(String[] args) {
14        Scanner sc = new Scanner(System.in);
15
16        System.out.print("Enter a number: ");
17        int n = sc.nextInt();
18
19        // Calculate and display sum of digits
20        int result = sum(n);
21        System.out.println("Sum of digits: " + result);
22
23        sc.close();
24    }
25
26 }
```

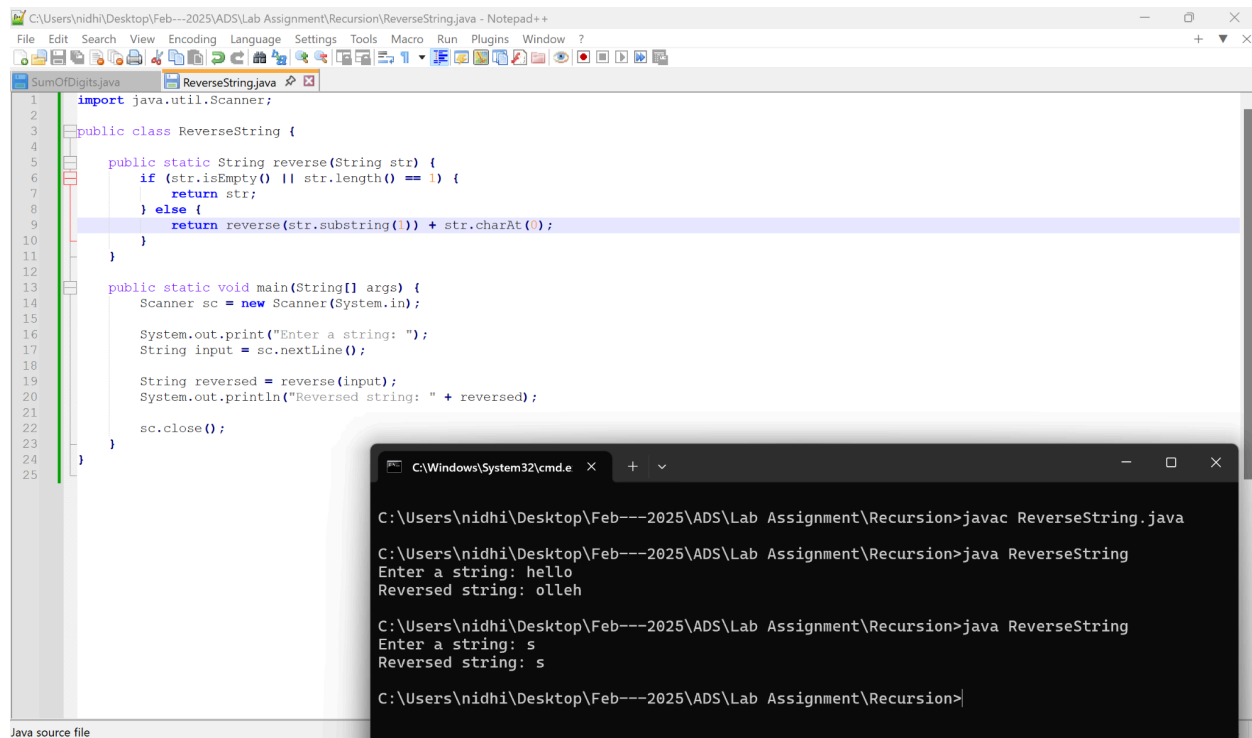
```
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac SumOfDigits.java
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java SumOfDigits
Enter a number: 1
Sum of digits: 1
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java SumOfDigits
Enter a number: 987
Sum of digits: 24
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>
```

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

$\text{sum}(987) = \text{sum}(98) + 7$
 $\text{sum}(98) = \text{sum}(9) + 8$
 $\text{sum}(9) = 9$ (Base case)

$\text{sum}(9) = 9$
 $\text{sum}(98) = 9 + 8 = 17$
 $\text{sum}(987) = 17 + 7 = 24$

Que 4 : Reverse The String



```
1 import java.util.Scanner;
2
3 public class ReverseString {
4
5     public static String reverse(String str) {
6         if (str.isEmpty() || str.length() == 1) {
7             return str;
8         } else {
9             return reverse(str.substring(1)) + str.charAt(0);
10        }
11    }
12
13    public static void main(String[] args) {
14        Scanner sc = new Scanner(System.in);
15
16        System.out.print("Enter a string: ");
17        String input = sc.nextLine();
18
19        String reversed = reverse(input);
20        System.out.println("Reversed string: " + reversed);
21
22        sc.close();
23    }
24
25 }
```

```
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac ReverseString.java
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java ReverseString
Enter a string: hello
Reversed string: olleh

C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java ReverseString
Enter a string: s
Reversed string: s

C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>
```

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.

$\text{reverse}(\text{"hello"}) = \text{reverse}(\text{"ello"}) + \text{'h'}$

$\text{reverse}(\text{"ello"}) = \text{reverse}(\text{"llo"}) + \text{'e'}$

$\text{reverse}(\text{"llo"}) = \text{reverse}(\text{"lo"}) + \text{'l'}$

$\text{reverse}(\text{"lo"}) = \text{reverse}(\text{"o"}) + \text{'l'}$

$\text{reverse}(\text{"o"}) = \text{reverse}(\text{""}) + \text{'o'}$ (Base case reached)

Returning from recursion:

$\text{reverse}(\text{"o"}) = \text{"o"}$

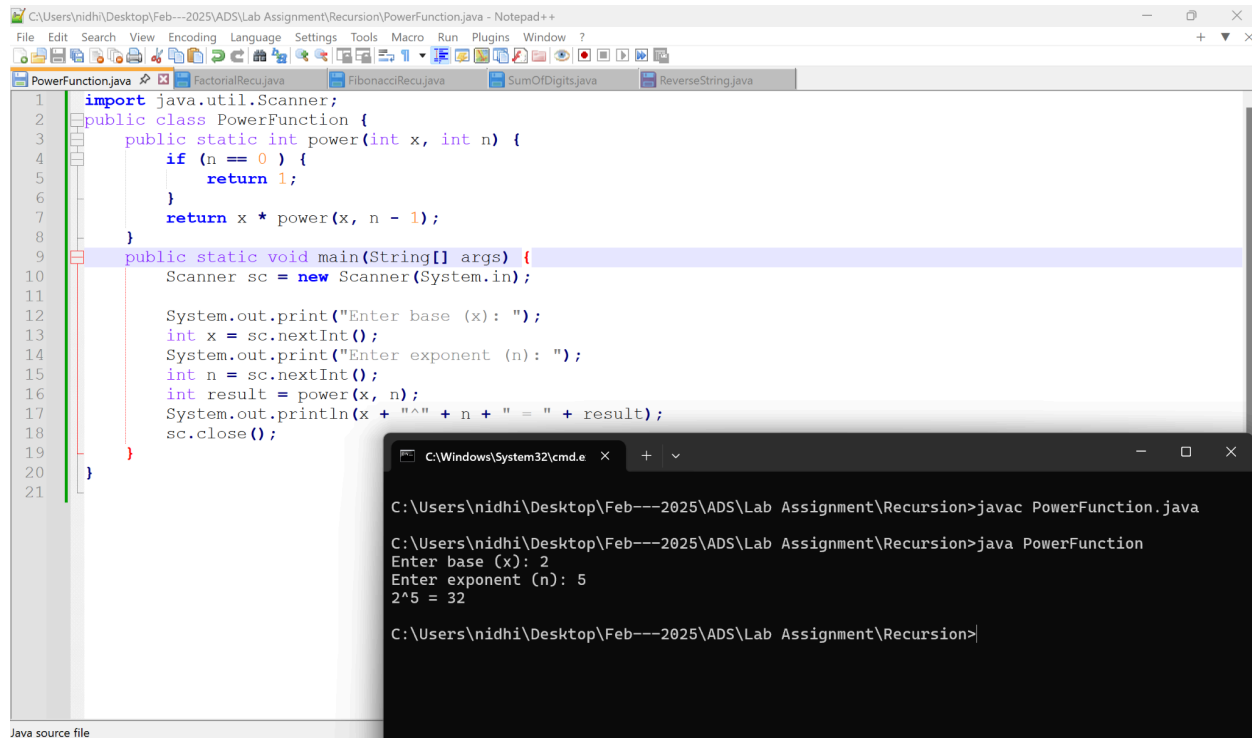
$\text{reverse}(\text{"lo"}) = \text{"o"} + \text{"l"} = \text{"ol"}$

$\text{reverse}(\text{"llo"}) = \text{"ol"} + \text{"l"} = \text{"oll"}$

$\text{reverse}(\text{"ello"}) = \text{"oll"} + \text{"e"} = \text{"olle"}$

$\text{reverse}(\text{"hello"}) = \text{"olle"} + \text{"h"} = \text{"olleh"}$

Que 5 : Power Function



The screenshot shows a Java IDE with the file `PowerFunction.java` open. The code defines a recursive power function. Below the code, a command prompt window shows the compilation and execution of the program, demonstrating the calculation of $2^5 = 32$.

```
1 import java.util.Scanner;
2 public class PowerFunction {
3     public static int power(int x, int n) {
4         if (n == 0) {
5             return 1;
6         }
7         return x * power(x, n - 1);
8     }
9     public static void main(String[] args) {
10        Scanner sc = new Scanner(System.in);
11
12        System.out.print("Enter base (x): ");
13        int x = sc.nextInt();
14        System.out.print("Enter exponent (n): ");
15        int n = sc.nextInt();
16        int result = power(x, n);
17        System.out.println(x + "^" + n + " = " + result);
18        sc.close();
19    }
20 }
21 }
```

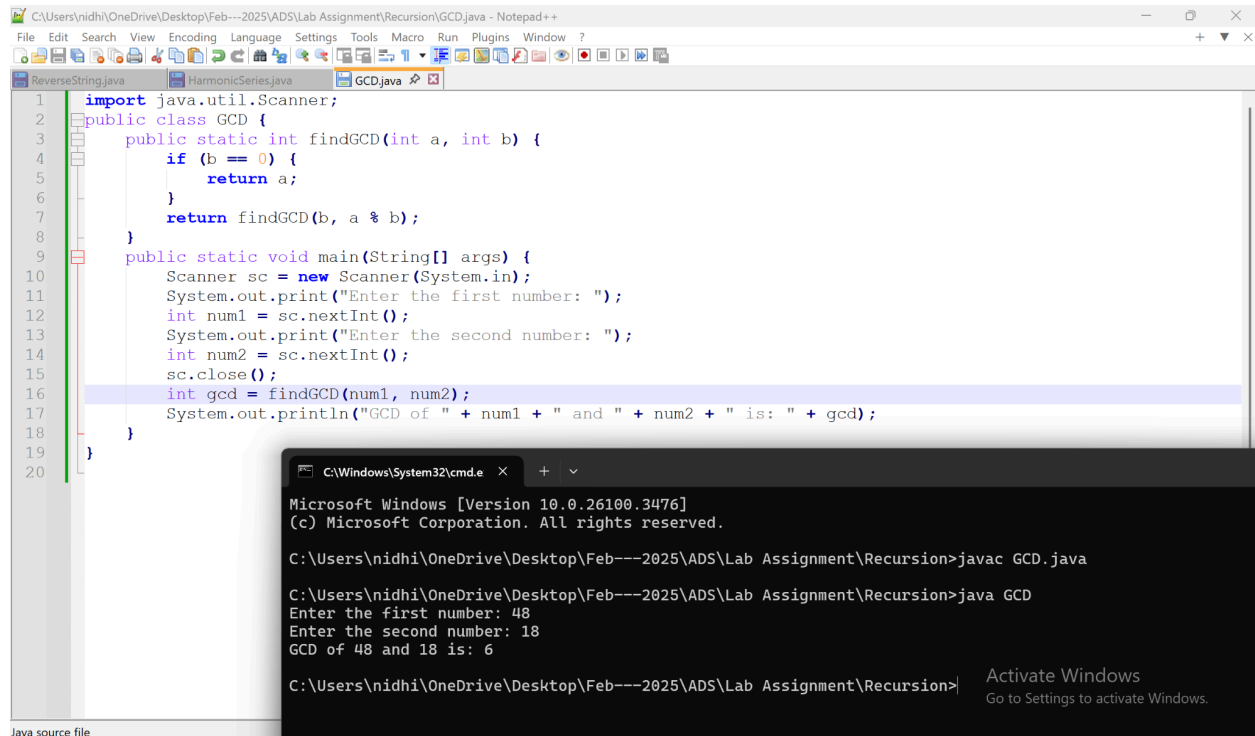
```
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac PowerFunction.java
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java PowerFunction
Enter base (x): 2
Enter exponent (n): 5
2^5 = 32
C:\Users\nidhi\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>
```

$\text{power}(2, 5) = 2 * \text{power}(2, 4)$
 $\text{power}(2, 4) = 2 * \text{power}(2, 3)$
 $\text{power}(2, 3) = 2 * \text{power}(2, 2)$
 $\text{power}(2, 2) = 2 * \text{power}(2, 1)$
 $\text{power}(2, 1) = 2 * \text{power}(2, 0)$
 $\text{power}(2, 0) = 1$ (Base case)

Returning from recursion:

$\text{power}(2, 1) = 2 * 1 = 2$
 $\text{power}(2, 2) = 2 * 2 = 4$
 $\text{power}(2, 3) = 2 * 4 = 8$
 $\text{power}(2, 4) = 2 * 8 = 16$
 $\text{power}(2, 5) = 2 * 16 = 32$

Que 6 : GCD using Recursive



```
1 import java.util.Scanner;
2 public class GCD {
3     public static int findGCD(int a, int b) {
4         if (b == 0) {
5             return a;
6         }
7         return findGCD(b, a % b);
8     }
9     public static void main(String[] args) {
10        Scanner sc = new Scanner(System.in);
11        System.out.print("Enter the first number: ");
12        int num1 = sc.nextInt();
13        System.out.print("Enter the second number: ");
14        int num2 = sc.nextInt();
15        sc.close();
16        int gcd = findGCD(num1, num2);
17        System.out.println("GCD of " + num1 + " and " + num2 + " is: " + gcd);
18    }
19 }
20 }
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.26100.3476]
(c) Microsoft Corporation. All rights reserved.

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

C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java GCD
Enter the first number: 48
Enter the second number: 18
GCD of 48 and 18 is: 6

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

Explanation:

1. Base Case: If $b == 0$, return a (GCD found)
2. Recursive Case: Call $\text{findGCD}(b, a \% b)$, reducing the problem size
3. Example Calculation for (48, 18):
 $\text{findGCD}(48, 18) = \text{findGCD}(18, 48 \% 18) = \text{findGCD}(18, 12)$
 $\text{findGCD}(18, 12) = \text{findGCD}(12, 18 \% 12) = \text{findGCD}(12, 6)$
 $\text{findGCD}(12, 6) = \text{findGCD}(6, 12 \% 6) = \text{findGCD}(6, 0)$
 $\text{findGCD}(6, 0)$ returns 6 (final answer).

Que 7 : Palindrome String

The screenshot shows a Notepad++ window with a Java file named `Palindrome.java`. The code defines a `Palindrome` class with a static method `isPalindrome` that uses recursion to check if a string is a palindrome. The `main` method uses a `Scanner` to take user input and prints the result.

```
1 import java.util.Scanner;
2 class Palindrome {
3     static boolean isPalindrome(String s) {
4         if (s.length() <= 1) {
5             return true;
6         }
7         if (s.charAt(0) == s.charAt(s.length() - 1)) {
8             return isPalindrome(s.substring(1, s.length() - 1));
9         }
10        return false;
11    }
12
13    public static void main(String[] args) {
14        Scanner sc = new Scanner(System.in);
15        System.out.println("Enter a string");
16        String str = sc.nextLine();
17        boolean result = isPalindrome(str);
18        System.out.println("Is the string: " + str);
19        System.out.println("String is palindrome: " + result);
20    }
21 }
```

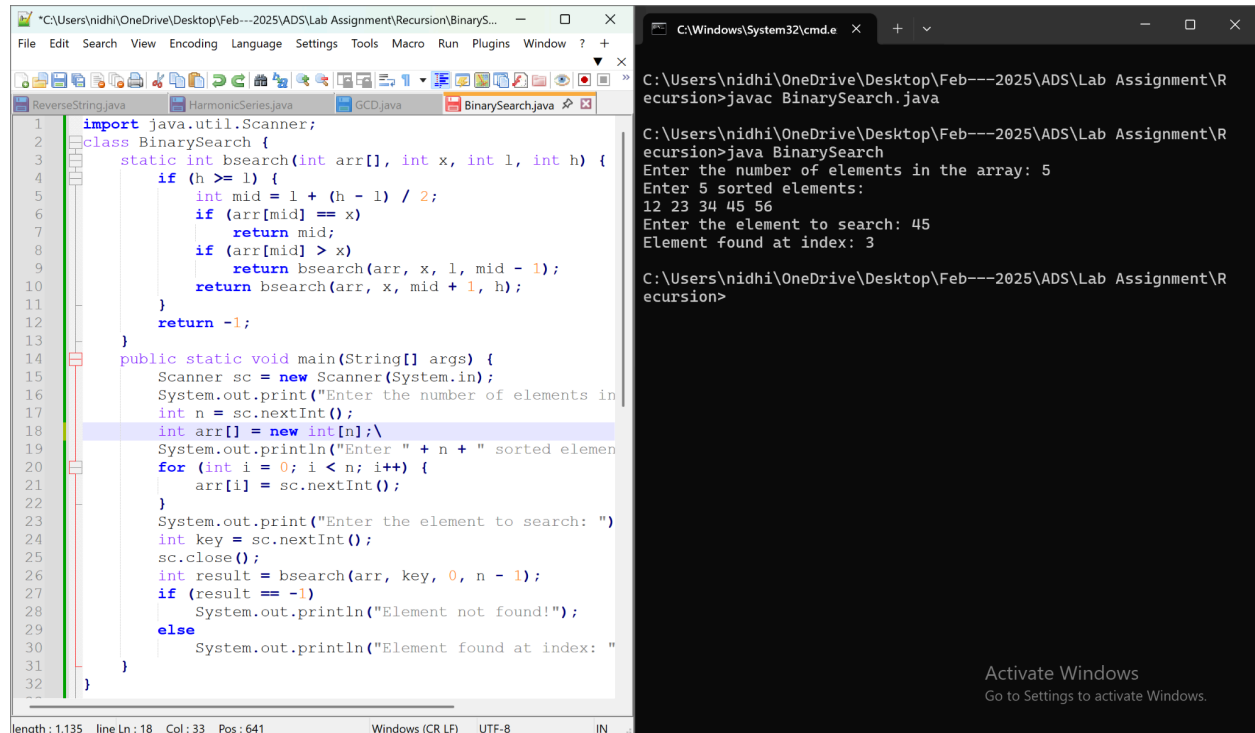
Below the editor, a Windows command prompt shows the execution of the program:

```
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



The image shows a Java IDE on the left and a Windows command prompt on the right. The IDE displays the source code for a `BinarySearch` class. The code includes a recursive `bsearch` method and a `main` method that uses a `Scanner` to take user input for the number of elements, the array elements, and the search key. The command prompt shows the execution of the program, where the user enters 5 for the number of elements, 12 23 34 45 56 for the sorted elements, and 45 for the element to search. The output shows the element found at index 3.

```
1 import java.util.Scanner;
2 class BinarySearch {
3     static int bsearch(int arr[], int x, int l, int h) {
4         if (h >= l) {
5             int mid = l + (h - l) / 2;
6             if (arr[mid] == x)
7                 return mid;
8             if (arr[mid] > x)
9                 return bsearch(arr, x, l, mid - 1);
10            return bsearch(arr, x, mid + 1, h);
11        }
12        return -1;
13    }
14    public static void main(String[] args) {
15        Scanner sc = new Scanner(System.in);
16        System.out.print("Enter the number of elements in");
17        int n = sc.nextInt();
18        int arr[] = new int[n];
19        System.out.println("Enter " + n + " sorted elements");
20        for (int i = 0; i < n; i++) {
21            arr[i] = sc.nextInt();
22        }
23        System.out.print("Enter the element to search: ");
24        int key = sc.nextInt();
25        sc.close();
26        int result = bsearch(arr, key, 0, n - 1);
27        if (result == -1)
28            System.out.println("Element not found!");
29        else
30            System.out.println("Element found at index: ");
31    }
32 }
```

```
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.

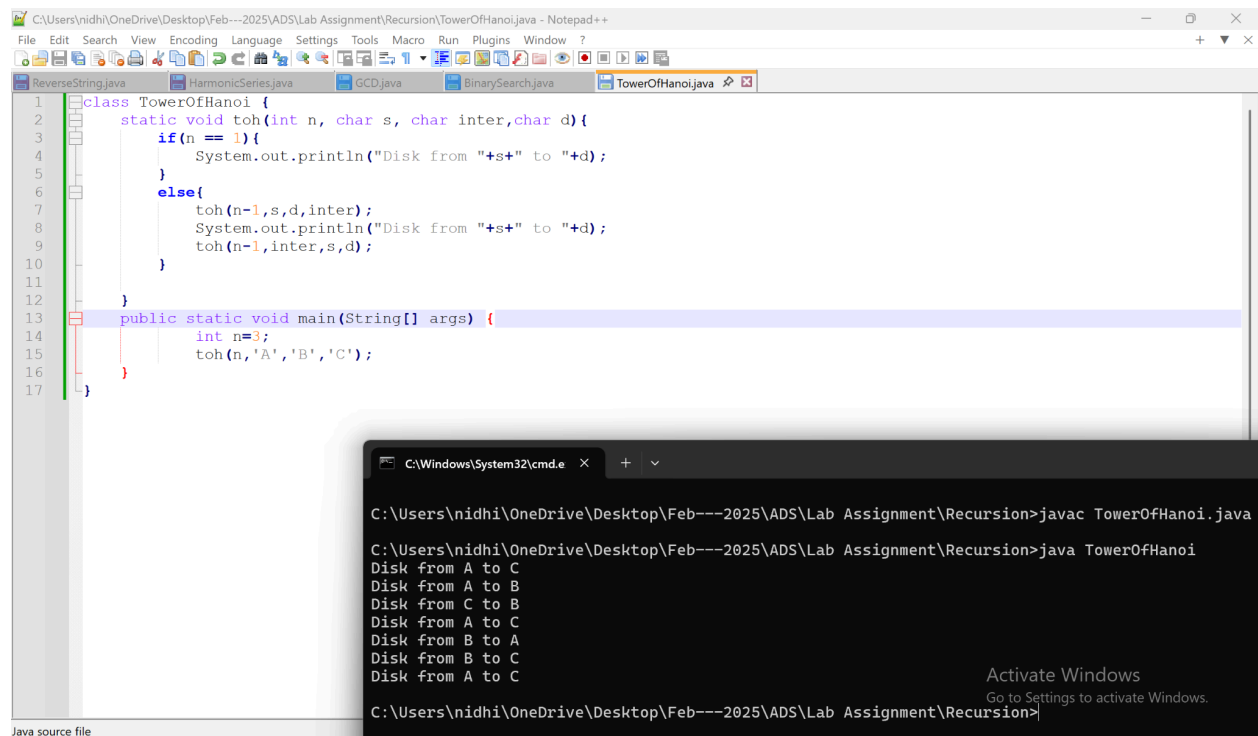
If `arr[mid] == key`, return index.

If `arr[mid] > key`, search left half.

Else, search right half.

If `l > h`, return -1 (not found).

Que 9 : Tower of Hanoi



```
1 class TowerOfHanoi {
2     static void toh(int n, char s, char inter, char d) {
3         if (n == 1) {
4             System.out.println("Disk from "+s+" to "+d);
5         }
6         else {
7             toh(n-1, s, d, inter);
8             System.out.println("Disk from "+s+" to "+d);
9             toh(n-1, inter, s, d);
10        }
11    }
12
13    public static void main(String[] args) {
14        int n=3;
15        toh('A', 'B', 'C');
16    }
17 }
```

```
C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>javac TowerOfHanoi.java
C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>java TowerOfHanoi
Disk from A to C
Disk from A to B
Disk from C to B
Disk from A to C
Disk from B to A
Disk from B to C
Disk from A to C
C:\Users\nidhi\OneDrive\Desktop\Feb---2025\ADS\Lab Assignment\Recursion>
```

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