**C-DAC Mumbai Date 25/09/2024**

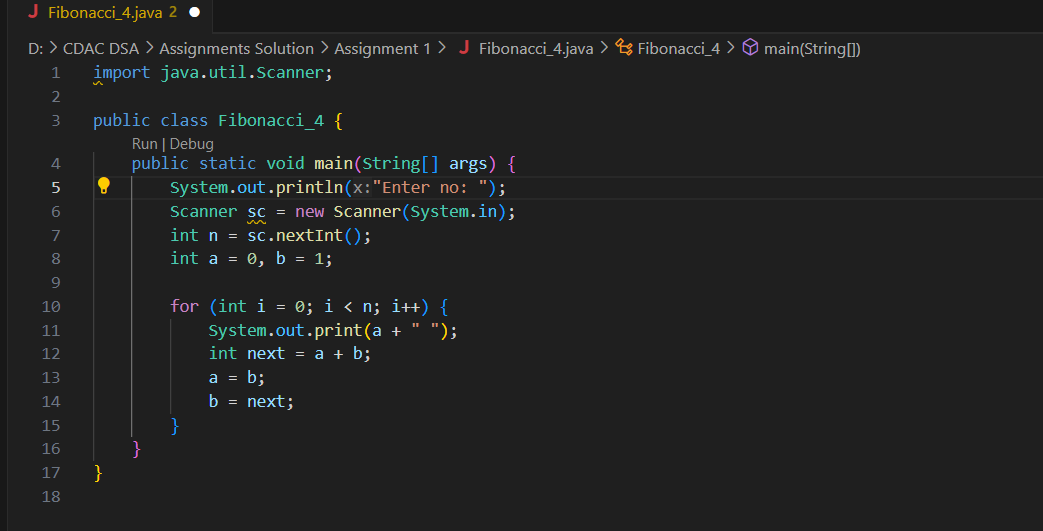
**Subject: Algorithm and Data Structure**

**Assignment 1**

4. Fibonacci Series

Problem: Write a Java program to print the first n numbers in the Fibonacci series.

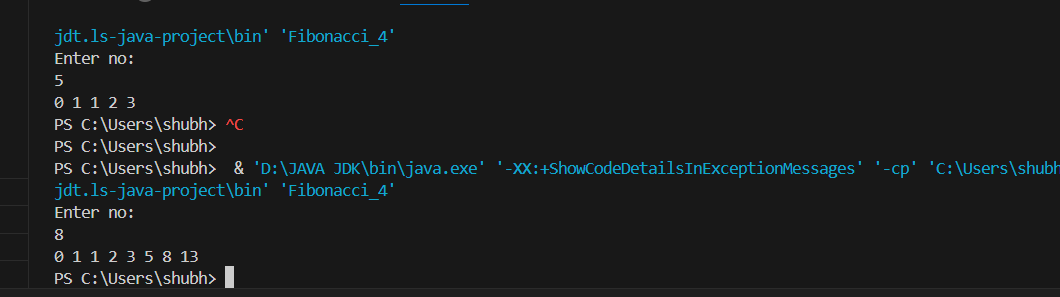
Program



Flowchart

1. **Start**:
   * The program starts.
2. **Ask for Input**:
   * The program asks the user to enter a number (n).
3. **Get Input**:
   * The user enters a number, and the program reads it.
4. **Set Initial Values**:
   * Set the first two numbers of the Fibonacci sequence: a = 0 and b = 1.
5. **Loop Start**:
   * The program starts a loop that will run n times.
6. **Print the Number**:
   * The program prints the current value of a.
7. **Calculate the Next Fibonacci Number**:
   * Calculate the next number in the Fibonacci sequence by adding a + b.
8. **Update Values**:
   * Set a to the value of b, and set b to the new Fibonacci number (the one you just calculated).
9. **Repeat**:
   * The loop repeats until the program has printed n Fibonacci numbers.
10. **End**:
    * Once the loop finishes, the program ends.

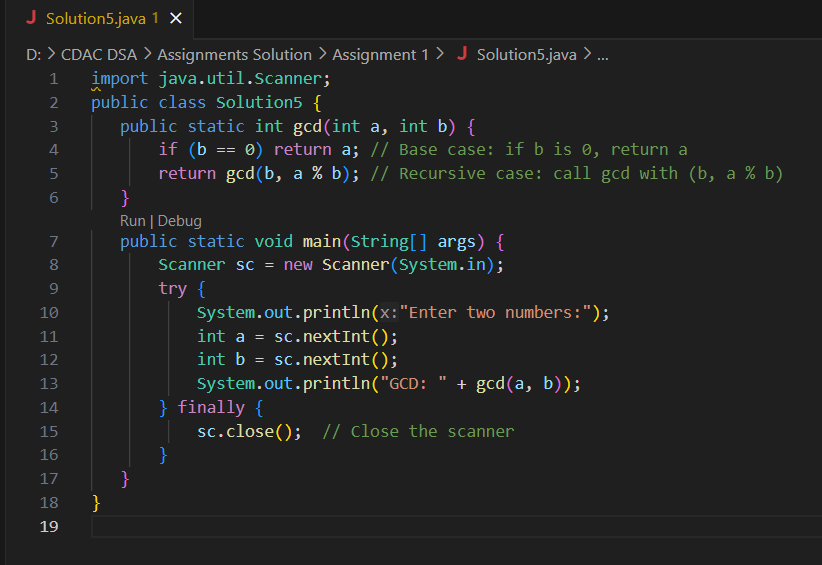
Output



Time Complexity: O(n)

Space Complexity: O(1)

**Q5. Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.**

****

Flowchart

Start

Ask user for two numbers (a, b)

Read a and b

Is b = 0?

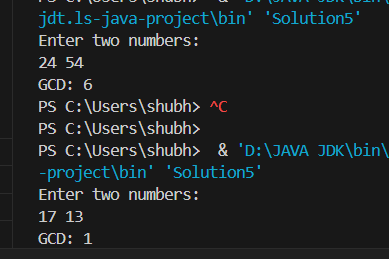
* Yes: Return a (GCD)
* No: Call gcd(b, a % b)

Display GCD

Close Scanner

End

Output



**Time Complexity**: O(log(min(a, b)))

**Space Complexity**: O(1)

1. Armstrong Number

Problem: Write a Java program to check if a given number is an Armstrong number.



**Flowchart:**

1.Start

      2.Input the number

      3.Store the number in a variable (original)

      4. Count number of digits

      5. Initialize sum to 0

      6. For each digit of given number :

              > Extract the last digit

              > Raise the digit to the power of no of digits

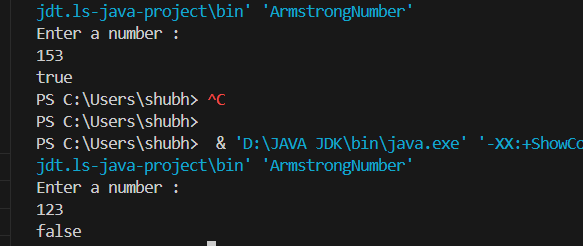
              > Add result to (sum)

              > Remove the last digit from the number

     7.If sum is equal to original,return TRUE otherwise return FALSE

     8.End

Output



**Time Complexity :**

    >> O(d),where d is no of digits in the input number.

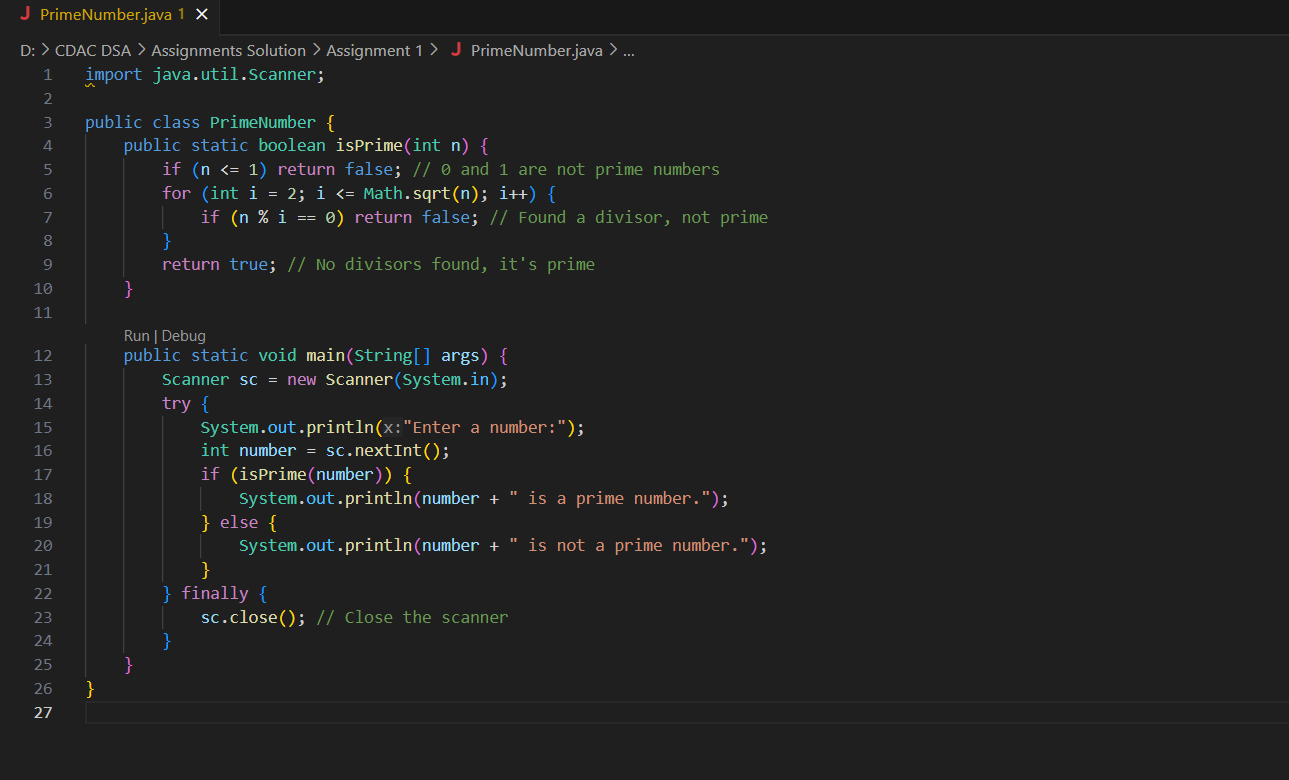
         The program extracts each digit,raises it to the power of d , and sums the result.

**Space Complexity :**

   >> O(1) : only few integer variables are used to store the result

2. Prime Number

Problem: Write a Java program to check if a given number is prime.



Flowchart

Start

Ask user for a number (n)

Read n

Is n <= 1?

* Yes: Print "n is not a prime number." → Go to Step 7
* No: Proceed to Step 5

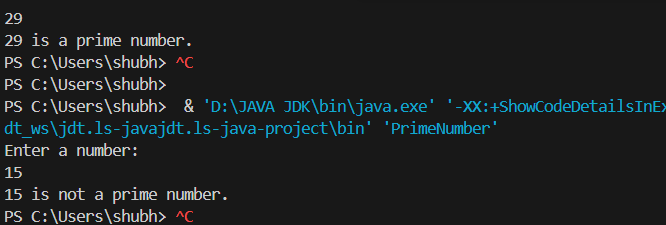
Set i = 2

Is i <= √n?

* Yes:
  + Is n % i == 0?
    - Yes: Print "n is not a prime number." → Go to Step 7
    - No: Increment i by 1 → Go back to Step 6
* No: Print "n is a prime number."

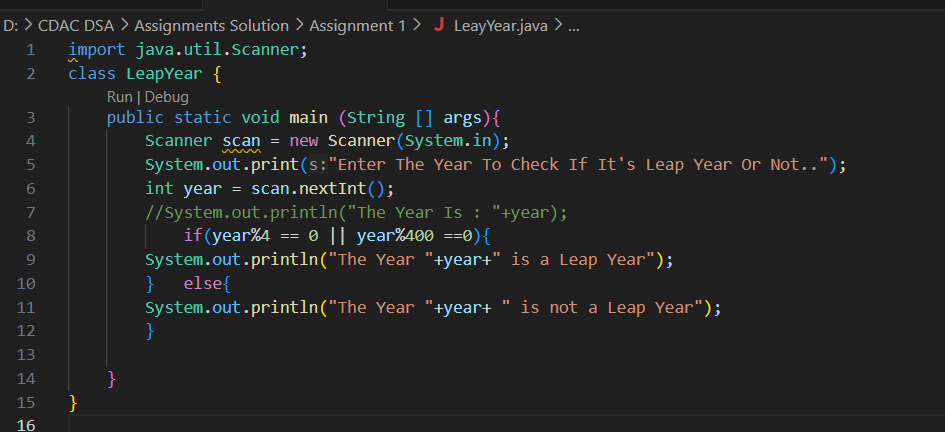
End

Output

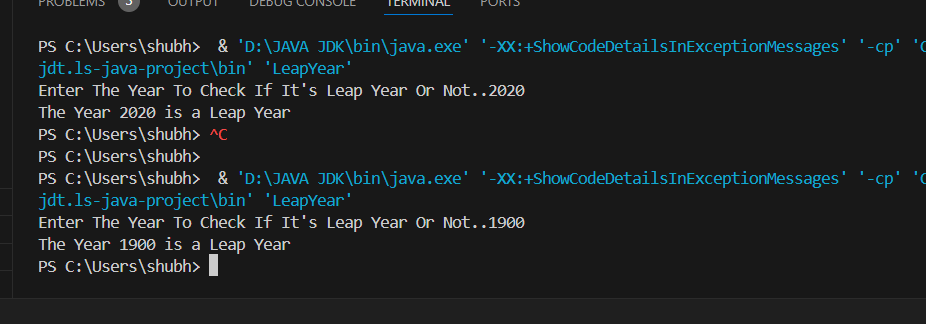


10. Leap Year

Problem: Write a Java program to check if a given year is a leap year.



OUTPUT



FLOWCHART

**Flowchart:  
Start:** Begin the algorithm.

**Input Year:** Prompt the user to enter a year and read the input value into the variable year.

**Check Conditions:**

**Condition 1:** Check if year is divisible by 4.

If true, proceed to Condition 2.

If false, go to Step 5 (not a leap year).

**Condition 2:** If false, go to next condition (it is a leap year).

**Condition 3:** Check if year is divisible by 400.

If true, go to Step 4 (it is a leap year).

If false, go to Step 5 (not a leap year).

**Output Leap Year:** Print "The Year year is a Leap Year."

**Output Not Leap Year:** Print "The Year year is not a Leap Year."

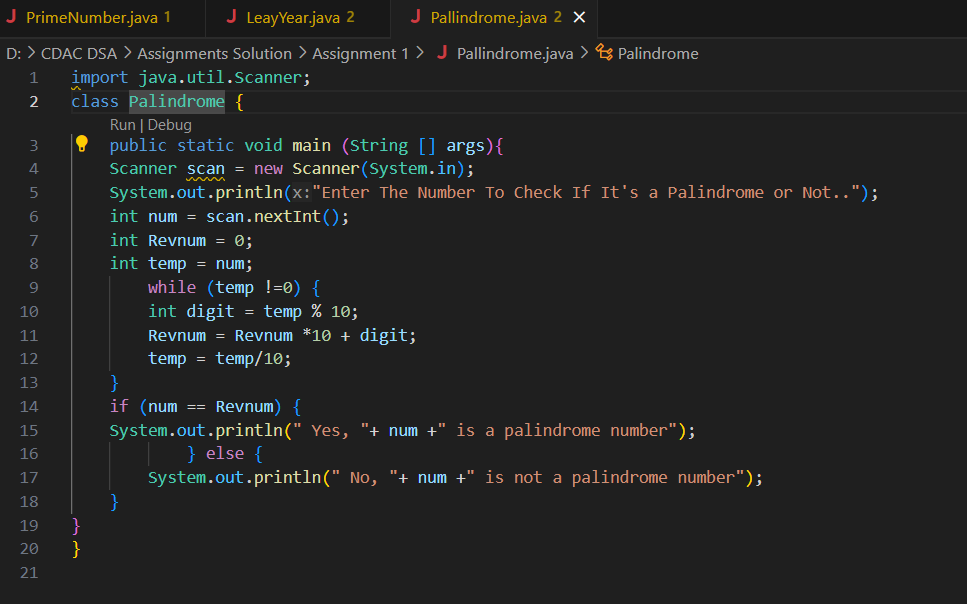
**End:** Terminate the algorithm.

Time Complexity : O(1)

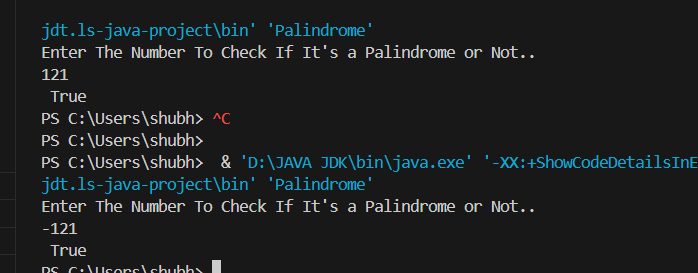
Space Complexity : O(1)

9. Integer Palindrome

Problem: Write a Java program to check if a given integer is a palindrome.



OUTPUT



Time Complexity : *O*(log10)

Space Complexity : O(1)

**Flowchart:**

**Start:** Begin the algorithm.

**Input Number:** Prompt the user to enter a number and read the input value into the variable num.

**Initialize Variables:**

Set Revnum to 0 (this will hold the reversed number).

Set temp to num (this will be used to extract digits).

**Reverse the Number:**

While temp is not equal to 0, repeat the following steps:

Extract the last digit of temp using digit = temp % 10.

Update Revnum by multiplying it by 10 and adding the extracted digit: Revnum = Revnum \* 10 + digit.

Remove the last digit from temp by performing integer division by 10: temp = temp / 10.

Check for Palindrome:

If num is equal to Revnum, then:

Print "Yes, num is a palindrome number."

**Otherwise:**

Print "No, num is not a palindrome number."

**End:** Terminate the algorithm.