**Name :- SACHIN HIMMAT AHIRE**

**PRN :- 240340520073**

**1.Check the given number is EVEN or ODD.**

**package** progs;

**import** java.util.Scanner;

**public** **class** CheckEvenOdd {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

**if** (number % 2 == 0) {

System.***out***.println(number + " is even.");

} **else** {

System.***out***.println(number + " is odd.");

}

scanner.close();

}

}

**Output:**

Enter a number: 5

5 is odd.

Enter a number: 4

4 is even.

/\*Start: Begin the process.

Prompt user to enter a number: Ask the user to input a number.

Read the number: Capture the user's input.

Is the number divisible by 2?: Check if the number modulo 2 is equal to 0.

Yes: If true, proceed to the next step.

No: If false, proceed to the step after the next one.

Print "The number is even": Output that the number is even.

Print "The number is odd": Output that the number is odd.

End: Terminate the process.\*/

**2. Write a Java Program to find the Factorialof given number.**

**package** progs;

**import** java.util.Scanner;

**public** **class** PrintFactorsOfNum {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

System.***out***.println("The factors of " + number + " are:");

**for** (**int** i = 1; i <= number; i++) {

**if** (number % i == 0) {

System.***out***.println(i);

}

}

scanner.close();

}

}

**Output:**

Enter a number: 5

The factors of 5 are:

1

5

/\*

Import the Scanner class: This allows us to take input from the user.

Create the main class: Define the main class FactorPrinter.

Create the main method: Define the main method to take user input and print the factors.

Prompt the user to enter a number: Display a message asking the user to input a number.

Read the number: Capture the user's input and store it in a variable number.

Find and print all factors:

Loop from 1 to the given number.

Check if the current loop variable i is a factor of the number (number % i == 0).

If it is, print i.

Close the Scanner: This releases the resource associated with the Scanner object

\*/

**3. Find the Factorialof a number using Recursion.**

**package** progs;

**import** java.util.Scanner;

//FactorialUsingRecursion

**public** **class** FactOfNumRecursion {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

**long** factorial = *calculateFactorial*(number);

System.***out***.println("The factorial of " + number + " is " + factorial);

scanner.close();

}

**public** **static** **long** calculateFactorial(**int** n) {

**if** (n == 0) {

**return** 1;

} **else** {

**return** n \* *calculateFactorial*(n - 1);

}

}

}

**Output:**

Enter a number: 5

The factorial of 5 is 120

/\*

Start

Input: Prompt the user to enter a number.

Read: Capture the user's input and store it in a variable number.

Call the recursive method: Calculate the factorial of number using the calculateFactorial method.

Print the result: Display the calculated factorial.

End

\*/

**4. Swap two numberswithout using third variable approach 1.**

**package** progs;

**import** java.util.Scanner;

**public** **class** SwapNum1 {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number: ");

**int** a = scanner.nextInt();

System.***out***.print("Enter the second number: ");

**int** b = scanner.nextInt();

a = a + b;

b = a - b;

a = a - b;

System.***out***.println("After swapping: a = " + a + ", b = " + b);

scanner.close();

}

}

**Output:**

Enter the first number: 5

Enter the second number: 6

After swapping: a = 6, b = 5

/\*

Start

Input: Read two numbers, a and b.

Step 1: Set a = a + b.

Step 2: Set b = a - b (Now b holds the original value of a).

Step 3: Set a = a - b (Now a holds the original value of b).

Output: Print the swapped values of a and b.

End

\*/

**5. Swap two numberswithout using third variable approach 2.**

**package** progs;

**import** java.util.Scanner;

// swap using bitwise operator SwapUsingXOR

**public** **class** SwapNum2 {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number: ");

**int** a = scanner.nextInt();

System.***out***.print("Enter the second number: ");

**int** b = scanner.nextInt();

a = a ^ b;

b = a ^ b;

a = a ^ b;

System.***out***.println("After swapping: a = " + a + ", b = " + b);

scanner.close();

}

}

**Output:**

Enter the first number: 5

Enter the second number: 6

After swapping: a = 6, b = 5

Start

Input: Read two numbers, a and b.

Step 1: Set a = a ^ b.

Step 2: Set b = a ^ b (Now b holds the original value of a).

Step 3: Set a = a ^ b (Now a holds the original value of b).

Output: Print the swapped values of a and b.

End

**6. Swap two numberswithout using third variable approach 3.**

**package** progs;

**import** java.util.Scanner;

//swap numbers with division and multiplication

**public** **class** SwapNum3 {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number: ");

**int** a = scanner.nextInt();

System.***out***.print("Enter the second number: ");

**int** b = scanner.nextInt();

// Swapping without using a third variable

a = a \* b;

b = a / b;

a = a / b;

System.***out***.println("After swapping: a = " + a + ", b = " + b);

scanner.close();

}

}

**Output:**

Enter the first number: 10

Enter the second number: 20

After swapping: a = 20, b = 10

/\*

Start

Input: Read two numbers, a and b.

Step 1: Set a = a \* b.

Step 2: Set b = a / b (Now b holds the original value of a).

Step 3: Set a = a / b (Now a holds the original value of b).

Output: Print the swapped values of a and b.

End

\*/

**7. How to check the given number is Positive or Negativein Java?**

**package** progs;

**import** java.util.Scanner;

**public** **class** CheckNumPosNeg {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**double** number = scanner.nextDouble();

**if** (number > 0) {

System.***out***.println(number + " is positive.");

} **else** **if** (number < 0) {

System.***out***.println(number + " is negative.");

} **else** {

System.***out***.println(number + " is neither positive nor negative.");

}

}

}

**Output:**

Enter a number: 7

7.0 is positive.

Enter a number: -7

-7.0 is negative.

Start: Begin the process.

Prompt user to enter a number: Display a message asking the user to input a number.

Read the number: Capture the user's input and store it in a variable number.

Check if the number is positive:

If number > 0:

Print "number is positive."

Check if the number is negative:

Else if number < 0:

Print "number is negative."

Check if the number is zero:

Else:

Print "number is neither positive nor negative."

End: Terminate the process.

**8. Write a Java Program to find whether given number is Leap year or NOT?**

**package** progs;

**import** java.util.Scanner;

**public** **class** LeapYearNumber {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a year: ");

**int** year = scanner.nextInt();

**if** (*isLeapYear*(year)) {

System.***out***.println(year + " is a leap year.");

} **else** {

System.***out***.println(year + " is not a leap year.");

}

scanner.close();

}

**public** **static** **boolean** isLeapYear(**int** year) {

**if** (year % 4 == 0) {

**if** (year % 100 == 0) {

**if** (year % 400 == 0) {

**return** **true**; // Divisible by 400

} **else** {

**return** **false**; // Divisible by 100 but not by 400

}

} **else** {

**return** **true**; // Divisible by 4 but not by 100

}

} **else** {

**return** **false**; // Not divisible by 4

}

}

}

/\*

**Output:**

Enter a year: 1992

1992 is a leap year.

Enter a year: 1993

1993 is not a leap year.

Start

Input: Prompt the user to enter a year.

Read: Capture the user's input and store it in a variable year.

Check if the year is a leap year:

If year % 4 == 0:

If year % 100 == 0:

If year % 400 == 0:

Output: Print "year is a leap year."

Else:

Output: Print "year is not a leap year."

Else:

Output: Print "year is a leap year."

Else:

Output: Print "year is not a leap year."

End

\*/

**10. Write a Java Program to print the digits of a Given Number.**

**package** progs;

**import** java.util.Scanner;

**public** **class** CountDigitInNum {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

// Convert the number to a string to handle each digit

String numberStr = Integer.*toString*(number);

System.***out***.println("The digits of " + number + " are:");

// Iterate through each character in the string and print it

**for** (**char** digit : numberStr.toCharArray()) {

System.***out***.println(digit);

}

scanner.close();

}

}

/\*

Start

Input: Prompt the user to enter a number.

Read: Capture the user's input and store it in a variable number.

Convert the number to a string: Use Integer.toString(number) to convert the number to a string.

Iterate through each character in the string:

For each character (digit) in the string representation, print it.

End

\*/

**11. Write a Java Program to print all the Factors of the Given number.**

**package** progs;

**import** java.util.Scanner;

**public** **class** PrintFactorsOfNum {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

System.***out***.println("The factors of " + number + " are:");

**for** (**int** i = 1; i <= number; i++) {

**if** (number % i == 0) {

System.***out***.println(i);

}

}

scanner.close();

}

}

**Output:**

Enter a number: 15

The factors of 15 are:

1

3

5

15

/\*

Import the Scanner class: This allows us to take input from the user.

Create the main class: Define the main class FactorPrinter.

Create the main method: Define the main method to take user input and print the factors.

Prompt the user to enter a number: Display a message asking the user to input a number.

Read the number: Capture the user's input and store it in a variable number.

Find and print all factors:

Loop from 1 to the given number.

Check if the current loop variable i is a factor of the number (number % i == 0).

If it is, print i.

Close the Scanner: This releases the resource associated with the Scanner object

\*/

**12. Write a Java Program to find sum of the digits of a given number.**

**package** progs;

**import** java.util.Scanner;

**public** **class** SumOfDigitsOfNum {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

// Variable to store the sum of digits

**int** sum = 0;

**int** temp = Math.*abs*(number); // Handle negative numbers

// Calculate the sum of digits

**while** (temp > 0) {

sum += temp % 10; // Add the last digit to the sum

temp /= 10; // Remove the last digit

}

System.***out***.println("The sum of the digits of " + number + " is " + sum);

scanner.close();

}

}

Output:

Enter a number: 55

The sum of the digits of 55 is 10

/\*

Start

Input: Prompt the user to enter a number.

Read: Capture the user's input and store it in a variable number.

Initialize sum to 0: This will hold the sum of the digits.

Handle negative numbers: Use Math.abs(number) to work with the absolute value.

Calculate sum of digits:

While the number is greater than 0:

Add the last digit to sum (using number % 10).

Remove the last digit from the number (using number /= 10).

Output: Print the sum of the digits.

End

\*/

/\*

Import the Scanner class: This allows us to take input from the user.

Create the main class: Define the main class SumOfDigits.

Create the main method: Define the main method to take user input and calculate the sum of digits.

Prompt the user to enter a number: Display a message asking the user to input a number.

Read the number: Capture the user's input and store it in a variable number.

Initialize sum to 0: This will hold the sum of the digits.

Handle negative numbers: Use Math.abs(number) to work with the absolute value of the number.

Calculate the sum of digits:

Use a while loop to extract and sum each digit of the number.

Inside the loop, add the last digit (temp % 10) to sum.

Remove the last digit by performing integer division (temp /= 10).

Print the result: Display the sum of the digits.

Close the Scanner: This releases the resource associated with the Scanner object.

\*/

**13. Write a Java Program to find the smallest of 3 numbers(a,b,c) without using < or > symbol?**

**package** progs;

**import** java.util.Scanner;

**public** **class** SmallestOf3Num {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number (a): ");

**int** a = scanner.nextInt();

System.***out***.print("Enter the second number (b): ");

**int** b = scanner.nextInt();

System.***out***.print("Enter the third number (c): ");

**int** c = scanner.nextInt();

// Find the smallest of the three numbers without using < or >

**int** smallest = *findSmallest*(a, b, c);

System.***out***.println("The smallest number is: " + smallest);

scanner.close();

}

**public** **static** **int** findSmallest(**int** a, **int** b, **int** c) {

// Use subtraction and addition to avoid < and >

**int** minAB = (a + b - Math.*abs*(a - b)) / 2;

**int** minABC = (minAB + c - Math.*abs*(minAB - c)) / 2;

**return** minABC;

}

}

**Output:**

Enter the first number (a): 8

Enter the second number (b): 6

Enter the third number (c): 9

The smallest number is: 6

/\*

Start

Input: Prompt the user to enter three numbers a, b, and c.

Read: Capture the user's input for a, b, and c.

Find the smallest:

Compute the smaller of a and b using (a + b - Math.abs(a - b)) / 2.

Compute the smallest of the result from the previous step and c using (minAB + c - Math.abs(minAB - c)) / 2.

Output: Print the smallest number.

End

\*/

/\*

Import the Scanner class: This allows us to take input from the user.

Create the main class: Define the main class SmallestNumberFinder.

Create the main method: Define the main method to take user input and determine the smallest number.

Prompt the user to enter three numbers: Display messages asking the user to input numbers a, b, and c.

Read the numbers: Capture the user’s input and store it in variables a, b, and c.

Find the smallest number:

Call the findSmallest method with a, b, and c.

The method calculates the smallest number using mathematical operations.

Print the result: Display the smallest number.

Close the Scanner: Release the resource associated with the Scanner object.

Define the findSmallest method:

Calculate the smaller of a and b using (a + b - Math.abs(a - b)) / 2.

Calculate the smallest of the result and c using the same method.

\*/

**14. How to add two numbers without using the arithmetic operators in Java?**

**package** progs;

**import** java.util.Scanner;

**public** **class** Add2NumWithoutArithmat {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number: ");

**int** num1 = scanner.nextInt();

System.***out***.print("Enter the second number: ");

**int** num2 = scanner.nextInt();

**int** sum = *add*(num1, num2);

System.***out***.println("The sum of " + num1 + " and " + num2 + " is " + sum);

scanner.close();

}

**public** **static** **int** add(**int** a, **int** b) {

// Loop until there is no carry

**while** (b != 0) {

// Carry now contains common set bits of a and b

**int** carry = a & b;

// Sum of bits of a and b where at least one of the bits is not set

a = a ^ b;

// Carry is shifted by one so that it can be added to a

b = carry << 1;

}

**return** a;

}

}

**Output:**

Enter the first number: 12

Enter the second number: 45

The sum of 12 and 45 is 57

/\*

Start

Input: Prompt the user to enter two numbers a and b.

Read: Capture the user's input for a and b.

Add without arithmetic operators:

While loop: Continue as long as b is not 0:

Compute carry as a & b.

Update a to be a ^ b.

Update b to be carry << 1.

Output: Print the sum.

End

\*/

/\*

Import the Scanner class: This allows us to take input from the user.

Create the main class: Define the main class AddWithoutArithmetic.

Create the main method: Define the main method to take user input and calculate the sum.

Prompt the user to enter two numbers: Display messages asking the user to input the numbers.

Read the numbers: Capture the user’s input and store it in variables num1 and num2.

Call the add method: Calculate the sum using bitwise operations.

Print the result: Display the result.

Close the Scanner: Release the resource associated with the Scanner object.

Define the add method:

Use a while loop to repeatedly add the carry to the sum until there is no carry left.

Compute the carry using a & b.

Update a to be the XOR of a and b.

Shift the carry left by one position and update b.

\*/

**15. Write a java program to Reverse a given number.**

**package** progs;

**import** java.util.Scanner;

**public** **class** ReverseNumber {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter a number: ");

**int** number = scanner.nextInt();

**int** reversedNumber = *reverse*(number);

System.***out***.println("The reversed number is: " + reversedNumber);

scanner.close();

}

**public** **static** **int** reverse(**int** number) {

**int** reversed = 0;

**int** originalNumber = number; // Save the original number for reference

**while** (number != 0) {

**int** digit = number % 10; // Extract the last digit

reversed = reversed \* 10 + digit; // Append the digit to the reversed number

number /= 10; // Remove the last digit from the original number

}

**return** reversed;

}

}

**Output:**

Enter a number: 459

The reversed number is: 954

/\*

Start

Input: Prompt the user to enter a number.

Read: Capture the user's input and store it in a variable number.

Reverse the number:

Initialize reversed to 0.

While number is not 0:

Extract the last digit using number % 10.

Update reversed to be reversed \* 10 + digit.

Update number to be number / 10.

Output: Print the reversed number.

End

\*/

**16. Write a Java Program to find GCD of two given numbers.**

**package** progs;

**import** java.util.Scanner;

**public** **class** GCDofNum {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number: ");

**int** num1 = scanner.nextInt();

System.***out***.print("Enter the second number: ");

**int** num2 = scanner.nextInt();

**int** gcd = *findGCD*(num1, num2);

System.***out***.println("The GCD of " + num1 + " and " + num2 + " is " + gcd);

scanner.close();

}

**public** **static** **int** findGCD(**int** a, **int** b) {

// Ensure both numbers are positive

a = Math.*abs*(a);

b = Math.*abs*(b);

// Euclidean algorithm

**while** (b != 0) {

**int** temp = b;

b = a % b;

a = temp;

}

**return** a;

}

}

**Output:**

Enter the first number: 55

Enter the second number: 66

The GCD of 55 and 66 is 11

/\*

Start

Input: Prompt the user to enter two numbers a and b.

Read: Capture the user's input for a and b.

Find GCD:

Ensure both numbers are positive.

While b is not 0:

Update b to a % b.

Update a to the previous value of b.

The GCD is the final value of a.

Output: Print the GCD.

End

\*/

/\*

Import the Scanner class: This allows us to take input from the user.

Create the main class: Define the main class GCDCalculator.

Create the main method: Define the main method to take user input and calculate the GCD.

Prompt the user to enter two numbers: Display messages asking the user to input the two numbers.

Read the numbers: Capture the user’s input and store it in variables num1 and num2.

Call the findGCD method: Calculate the GCD using the findGCD method.

Print the result: Display the calculated GCD.

Close the Scanner: Release the resource associated with the Scanner object.

Define the findGCD method:

Ensure both numbers are positive using Math.abs().

Use the Euclidean algorithm to find the GCD:

While b is not 0:

Store b in a temporary variable.

Update b to be a % b.

Update a to be the temporary variable.

Return a as the GCD.

\*/

**17. Write a java program to LCM of TWO given number.**

**package** progs;

**import** java.util.Scanner;

**public** **class** FindLCMof2Num {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the first number: ");

**int** num1 = scanner.nextInt();

System.***out***.print("Enter the second number: ");

**int** num2 = scanner.nextInt();

**int** lcm = *findLCM*(num1, num2);

System.***out***.println("The LCM of " + num1 + " and " + num2 + " is " + lcm);

scanner.close();

}

**public** **static** **int** findGCD(**int** a, **int** b) {

// Ensure both numbers are positive

a = Math.*abs*(a);

b = Math.*abs*(b);

// Euclidean algorithm to find GCD

**while** (b != 0) {

**int** temp = b;

b = a % b;

a = temp;

}

**return** a;

}

**public** **static** **int** findLCM(**int** a, **int** b) {

// Calculate LCM using the relationship with GCD

**return** Math.*abs*(a \* b) / *findGCD*(a, b);

}

}

**Output:**

Enter the first number: 8

Enter the second number: 6

The LCM of 8 and 6 is 24

/\*

Start

Input: Prompt the user to enter two numbers a and b.

Read: Capture the user's input for a and b.

Find GCD:

Ensure both numbers are positive.

While b is not 0:

Update b to a % b.

Update a to the previous value of b.

The GCD is the final value of a.

Calculate LCM

Output: Print the LCM.

End

\*/

Q1. Check the given number is EVEN or ODD.

import java.util.Scanner;

public class EvenOddCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter an integer: ");

int number = scanner.nextInt();

// Check if the number is even or odd

if (number % 2 == 0) {

System.out.println(number + " is even.");

} else {

System.out.println(number + " is odd.");

}

// Close the scanner

scanner.close();

}

}

2. Flow Chart

+---------------------------+

| Start |

+---------------------------+

|

v

+---------------------------+

| Input Integer |

+---------------------------+

|

v

+---------------------------+

| number % 2 == 0 ? |

+---------------------------+

/ \

/ \

v v

+----------------+ +-----------------+

| Print "Even" | | Print "Odd" |

+----------------+ +-----------------+

| |

v v

+---------------------------+

| End |

+---------------------------+

**4. Output**

The output of the program will vary based on the input. For example:

* **Input**: 5
  + **Output**: 5 is odd.
* **Input**: 12
  + **Output**: 12 is even.

**5. Time and Space Complexity**

* **Time Complexity**: O(1)
* **Space Complexity**: O(1)

2. Write a Java Program to find the Factorialof given number.

import java.util.Scanner;

public class FactorialCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a non-negative integer: ");

int number = scanner.nextInt();

// Check for non-negative input

if (number < 0) {

System.out.println("Factorial is not defined for negative numbers.");

} else {

// Calculate factorial

long factorial = 1;

for (int i = 1; i <= number; i++) {

factorial \*= i;

}

System.out.println("Factorial of " + number + " is " + factorial + ".");

}

// Close the scanner

scanner.close();

}

}

+---------------------------+

| Start |

+---------------------------+

|

v

+---------------------------+

| Input Integer |

+---------------------------+

|

v

+---------------------------+

| number < 0 ? |

+---------------------------+

/ \

/ \

v v

+----------------+ +-----------------+

| Print "Factorial| | Initialize |

| is not defined | | factorial = 1 |

| for negative | +-----------------+

| numbers." | |

+----------------+ v

+---------------------------+

| For i = 1 to number |

| factorial \*= i |

+---------------------------+

|

v

+---------------------------+

| Print "Factorial of |

| number is factorial." |

+---------------------------+

|

v

+---------------------------+

| End |

+---------------------------+

**Explanation**

1. **Input Integer**: The program prompts the user to enter a non-negative integer.
2. **Check Non-Negativity**: The program first checks if the input number is negative. If it is, it prints a message stating that the factorial is not defined for negative numbers.
3. **Calculate Factorial**:
   * If the number is non-negative, the program initializes a variable factorial to 1.
   * It then uses a for loop to iterate from 1 to the input number, multiplying factorial by the current loop index i on each iteration.
4. **Output Result**: After completing the loop, the program prints the computed factorial.
5. **Close Scanner**: The scanner is closed to release resources.

**4. Output**

The output of the program will depend on the user's input. For example:

* **Input**: 5
  + **Output**: Factorial of 5 is 120.
* **Input**: 0
  + **Output**: Factorial of 0 is 1.

**5. Time and Space Complexity**

* **Time Complexity**: O(n)
* **Space Complexity**: O(1)

Find the Factorial of a number using Recursion.

import java.util.Scanner;

public class RecursiveFactorial {

// Recursive method to calculate factorial

public static long factorial(int n) {

if (n == 0) {

return 1; // Base case: factorial of 0 is 1

} else {

return n \* factorial(n - 1); // Recursive case

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a non-negative integer: ");

int number = scanner.nextInt();

// Check for non-negative input

if (number < 0) {

System.out.println("Factorial is not defined for negative numbers.");

} else {

// Calculate factorial using recursive method

long result = factorial(number);

System.out.println("Factorial of " + number + " is " + result + ".");

}

// Close the scanner

scanner.close();

}

}

**FLOWCHART**:-

+---------------------------+

| Start |

+---------------------------+

|

v

+---------------------------+

| Input Integer |

+---------------------------+

|

v

+---------------------------+

| number < 0 ? |

+---------------------------+

/ \

/ \

v v

+----------------+ +-----------------+

| Print "Factorial| | Call factorial |

| is not defined | | method with n |

| for negative | | (recursive case)|

| numbers." | +-----------------+

+----------------+ |

v

+---------------------------+

| factorial(n) = n \* |

| factorial(n - 1) |

+---------------------------+

|

v

+---------------------------+

| Print "Factorial of |

| number is result." |

+---------------------------+

|

v

+---------------------------+

| End |

+---------------------------+

**5. Time and Space Complexity**

* **Time Complexity**: O(n)
* **Space Complexity**: O(n)

Swap two numberswithout using third variable approach 1.

public class SwapNumbers {

public static void main(String[] args) {

// Initial values

int a = 5;

int b = 10;

// Display initial values

System.out.println("Before swapping:");

System.out.println("a = " + a);

System.out.println("b = " + b);

// Swap the numbers using arithmetic operations

a = a + b; // Step 1

b = a - b; // Step 2

a = a - b; // Step 3

// Display values after swapping

System.out.println("After swapping:");

System.out.println("a = " + a);

System.out.println("b = " + b);

}

}

**OUTPUT :-**

Before swapping:

a = 5

b = 10

After swapping:

a = 10

b = 5

Swapping two numbers without using a third variable is a common problem in programming. It can be achieved through various methods. Here, I'll explain a method using arithmetic operations, which is one of the most commonly used techniques.

### Approach 1: Swapping Using Arithmetic Operations

In this approach, we use addition and subtraction to swap the values of two variables. Here’s a step-by-step explanation and corresponding Java code.

#### Explanation

1. **Initial Values**:
   * Let a and b be the two numbers you want to swap.
2. **Swap Process**:
   * **Step 1**: Add a and b and assign the result to a. a=a+ba = a + ba=a+b
   * **Step 2**: Subtract the new value of b from a to get the original value of a, and assign it to b. b=a−bb = a - bb=a−b
   * **Step 3**: Subtract the new value of b from a to get the original value of b, and assign it to a. a=a−ba = a - ba=a−b

After these operations, the values of a and b will be swapped.

#### Java Program

Here’s a Java program implementing the swapping of two numbers without using a third variable:

java

Copy code

public class SwapNumbers {

public static void main(String[] args) {

// Initial values

int a = 5;

int b = 10;

// Display initial values

System.out.println("Before swapping:");

System.out.println("a = " + a);

System.out.println("b = " + b);

// Swap the numbers using arithmetic operations

a = a + b; // Step 1

b = a - b; // Step 2

a = a - b; // Step 3

// Display values after swapping

System.out.println("After swapping:");

System.out.println("a = " + a);

System.out.println("b = " + b);

}

}

### Time and Space Complexity

* **Time Complexity**: O(1)
* **Space Complexity**: O(1)

26. Write a Java Program to print Perfect Numbersbetween 1 to 1000.

public class PerfectNumbers {

// Method to check if a number is perfect

public static boolean isPerfectNumber(int num) {

int sum = 0;

for (int i = 1; i <= num / 2; i++) {

if (num % i == 0) {

sum += i;

}

}

return sum == num;

}

public static void main(String[] args) {

System.out.println("Perfect numbers between 1 and 1000 are:");

// Check for perfect numbers between 1 and 1000

for (int i = 1; i <= 1000; i++) {

if (isPerfectNumber(i)) {

System.out.println(i);

}

}

}

}

### Output

Perfect numbers between 1 and 1000 are:

6

28

496

### Time and Space Complexity

* **Time Complexity**:
* **Space Complexity**:

Write a Java Program to check whether the given number is Perfect Numberor NOT.

import java.util.Scanner;

public class PerfectNumberChecker {

// Method to check if a number is perfect

public static boolean isPerfectNumber(int num) {

if (num <= 0) {

return false; // Perfect numbers must be positive

}

int sum = 0;

for (int i = 1; i <= num / 2; i++) {

if (num % i == 0) {

sum += i;

}

}

return sum == num;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a positive integer: ");

int number = scanner.nextInt();

// Check if the number is a perfect number

if (isPerfectNumber(number)) {

System.out.println(number + " is a perfect number.");

} else {

System.out.println(number + " is not a perfect number.");

}

// Close the scanner

scanner.close();

}

}

### Explanation

### Output

When you run the program and input a number, you will receive output indicating whether the number is perfect or not. For example:

* **Input**: 6
  + **Output**: 6 is a perfect number.
* **Input**: 12
  + **Output**: 12 is not a perfect number.

### Time and Space Complexity

* **Time Complexity**: O(n)
* **Space Complexity**: O(1)

Write a Java Program to print Armstrong Numbersbetween 1 to 1000.

public class ArmstrongNumbers {

// Method to check if a number is an Armstrong number

public static boolean isArmstrongNumber(int num) {

int originalNum = num;

int sum = 0;

int numberOfDigits = String.valueOf(num).length();

while (num > 0) {

int digit = num % 10;

sum += Math.pow(digit, numberOfDigits);

num /= 10;

}

return sum == originalNum;

}

public static void main(String[] args) {

System.out.println("Armstrong numbers between 1 and 1000 are:");

// Check for Armstrong numbers between 1 and 1000

for (int i = 1; i <= 1000; i++) {

if (isArmstrongNumber(i)) {

System.out.println(i);

}

}

}

}

OUTPUT :-

Armstrong numbers between 1 and 1000 are:

1

2

3

4

5

6

7

8

9

153

370

371

407

### Time and Space Complexity

* **Time Complexity**: O(n \* d)
* **Space Complexity**: O(1)

Write a Java Program to check whether the given number is Armstrong Numberor NOT.

import java.util.Scanner;

public class ArmstrongNumberChecker {

// Method to check if a number is an Armstrong number

public static boolean isArmstrongNumber(int num) {

int originalNum = num;

int sum = 0;

int numberOfDigits = String.valueOf(num).length();

while (num > 0) {

int digit = num % 10;

sum += Math.pow(digit, numberOfDigits);

num /= 10;

}

return sum == originalNum;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a positive integer: ");

int number = scanner.nextInt();

// Check if the number is an Armstrong number

if (isArmstrongNumber(number)) {

System.out.println(number + " is an Armstrong number.");

} else {

System.out.println(number + " is not an Armstrong number.");

}

// Close the scanner

scanner.close();

}

}

### Output

* **Input**: 153
  + **Output**: 153 is an Armstrong number.
* **Input**: 123
  + **Output**: 123 is not an Armstrong number.

### Time and Space Complexity

* **Time Complexity**: O(d)
* **Space Complexity**: O(1)

Write a Java Program to print Prime Numbers from 1 to N.

import java.util.Scanner;

public class PrimeNumbersPrinter {

// Method to check if a number is prime

public static boolean isPrime(int num) {

if (num <= 1) {

return false; // 0 and 1 are not prime numbers

}

if (num == 2) {

return true; // 2 is the only even prime number

}

if (num % 2 == 0) {

return false; // Other even numbers are not prime

}

// Check odd divisors from 3 to sqrt(num)

for (int i = 3; i <= Math.sqrt(num); i += 2) {

if (num % i == 0) {

return false;

}

}

return true;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a positive integer N: ");

int N = scanner.nextInt();

System.out.println("Prime numbers from 1 to " + N + " are:");

// Print prime numbers from 1 to N

for (int i = 2; i <= N; i++) {

if (isPrime(i)) {

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

}

}

System.out.println(); // Print newline after the list

// Close the scanner

scanner.close();

}

}

### Output

* **Input**: 20
  + **Output**: Prime numbers from 1 to 20 are: 2 3 5 7 11 13 17 19

### Time and Space Complexity

* **Time Complexity**: O(N \* sqrt(N))
* **Space Complexity**: O(1)

Write a Java Program to check whether the Given Number is Prime Number or NOT.

import java.util.Scanner;

public class PrimeNumberChecker {

// Method to check if a number is prime

public static boolean isPrime(int num) {

if (num <= 1) {

return false; // Numbers less than or equal to 1 are not prime

}

if (num == 2) {

return true; // 2 is the only even prime number

}

if (num % 2 == 0) {

return false; // Other even numbers are not prime

}

// Check for factors from 3 to sqrt(num)

for (int i = 3; i <= Math.sqrt(num); i += 2) {

if (num % i == 0) {

return false;

}

}

return true;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a positive integer: ");

int number = scanner.nextInt();

// Check if the number is prime

if (isPrime(number)) {

System.out.println(number + " is a prime number.");

} else {

System.out.println(number + " is not a prime number.");

}

// Close the scanner

scanner.close();

}

}

### Output

* **Input**: 29
  + **Output**: 29 is a prime number.
* **Input**: 30
  + **Output**: 30 is not a prime number.

### Time and Space Complexity

* **Time Complexity**: O(sqrt(n))
* **Space Complexity**: O(1)

Write a Java Program to print all the Prime Factorsof the Given Number.

import java.util.Scanner;

public class PrimeFactors {

// Method to print all prime factors of a given number

public static void printPrimeFactors(int num) {

// Handle the smallest prime number

while (num % 2 == 0) {

System.out.print(2 + " ");

num /= 2;

}

// Handle odd factors from 3 upwards

for (int i = 3; i <= Math.sqrt(num); i += 2) {

while (num % i == 0) {

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

num /= i;

}

}

// If num is still greater than 2, then it must be a prime number

if (num > 2) {

System.out.print(num + " ");

}

System.out.println(); // Print newline after the list

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a positive integer: ");

int number = scanner.nextInt();

// Print prime factors

System.out.print("Prime factors of " + number + " are: ");

printPrimeFactors(number);

// Close the scanner

scanner.close();

}

}

### Output

* **Input**: 315
  + **Output**: Prime factors of 315 are: 3 3 5 7
* **Input**: 60
  + **Output**: Prime factors of 60 are: 2 2 3 5

### Time and Space Complexity

* **Time Complexity**: O(sqrt(n))
* **Space Complexity**: O(1)

Check whether the Given Numberis a Palindrome or NOT.

import java.util.Scanner;

public class PalindromeNumberChecker {

// Method to check if a number is a palindrome

public static boolean isPalindrome(int num) {

int originalNum = num;

int reversedNum = 0;

// Reverse the number

while (num > 0) {

int digit = num % 10;

reversedNum = reversedNum \* 10 + digit;

num /= 10;

}

// Compare the reversed number with the original number

return originalNum == reversedNum;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter a positive integer: ");

int number = scanner.nextInt();

// Check if the number is a palindrome

if (isPalindrome(number)) {

System.out.println(number + " is a palindrome.");

} else {

System.out.println(number + " is not a palindrome.");

}

// Close the scanner

scanner.close();

}

}

### Output

* **Input**: 12321
  + **Output**: 12321 is a palindrome.
* **Input**: 12345
  + **Output**: 12345 is not a palindrome.

### Time and Space Complexity

* **Time Complexity**: O(d)
* **Space Complexity**: O(1)

Write a java program to LCM of TWO given number using Prime Factors method.

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

public class LCMUsingPrimeFactors {

// Method to get the prime factors of a number and their counts

public static Map<Integer, Integer> getPrimeFactors(int num) {

Map<Integer, Integer> factors = new HashMap<>();

// Handle the factor 2

while (num % 2 == 0) {

factors.put(2, factors.getOrDefault(2, 0) + 1);

num /= 2;

}

// Handle odd factors from 3 upwards

for (int i = 3; i <= Math.sqrt(num); i += 2) {

while (num % i == 0) {

factors.put(i, factors.getOrDefault(i, 0) + 1);

num /= i;

}

}

// If num is still greater than 2, then it must be a prime number

if (num > 2) {

factors.put(num, factors.getOrDefault(num, 0) + 1);

}

return factors;

}

// Method to calculate the LCM of two numbers using their prime factors

public static int calculateLCM(int num1, int num2) {

Map<Integer, Integer> factors1 = getPrimeFactors(num1);

Map<Integer, Integer> factors2 = getPrimeFactors(num2);

// Merge prime factors with the highest count

Map<Integer, Integer> lcmFactors = new HashMap<>();

for (Integer factor : factors1.keySet()) {

lcmFactors.put(factor, Math.max(factors1.get(factor), factors2.getOrDefault(factor, 0)));

}

for (Integer factor : factors2.keySet()) {

lcmFactors.put(factor, Math.max(lcmFactors.getOrDefault(factor, 0), factors2.get(factor)));

}

// Calculate the LCM by multiplying the highest powers of all prime factors

int lcm = 1;

for (Map.Entry<Integer, Integer> entry : lcmFactors.entrySet()) {

lcm \*= Math.pow(entry.getKey(), entry.getValue());

}

return lcm;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Prompt user for input

System.out.print("Enter the first positive integer: ");

int num1 = scanner.nextInt();

System.out.print("Enter the second positive integer: ");

int num2 = scanner.nextInt();

// Calculate and print the LCM

int lcm = calculateLCM(num1, num2);

System.out.println("The LCM of " + num1 + " and " + num2 + " is: " + lcm);

// Close the scanner

scanner.close();

}

}

### Output

* **Input**:

Enter the first positive integer: 12

Enter the second positive integer: 18

**Output**:

The LCM of 12 and 18 is: 36

### Time and Space Complexity

* **Time Complexity**: O(sqrt(n) + sqrt(m))
* **Space Complexity**: O(log(n) + log(m))