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

**Subject: Algorithm and Data Structure**

**Assignment 1**

**Solve the assignment with following thing to be added in each question.**

-Program

-Flow chart

-Explanation

-Output

-Time and Space complexity

1. **Armstrong Number**

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

Test Cases:

Input: 153

Output: true

Input: 123

Output: false

**Program:-**

class armstrong {

static int getArmstrongSum(int num, int num1) {

if(num == 0)

return 0;

int digit = num % 10;

return (int) Math.pow(digit, num1) + getArmstrongSum(num/10, num1);

}

static int num1(int num) {

int len = 0;

while (num!=0)

{

len++;

num = num/10;

}

return len;

}

public static void main(String[] args)

{

int num = 153;

int len = num1(num);

if (num == getArmstrongSum(num, len))

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

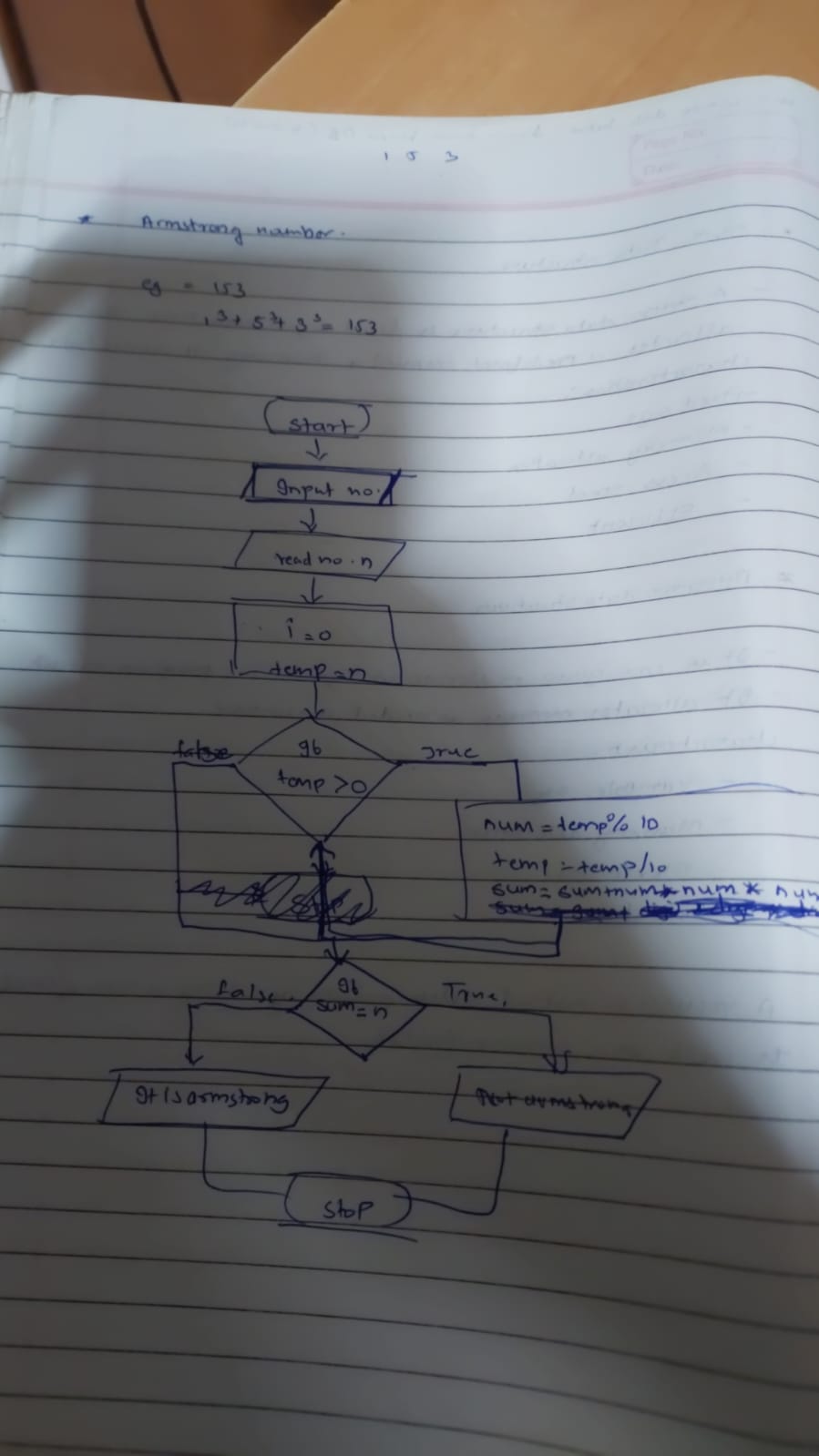
else

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

}

}

-Flow chart:-



**Explanation: -**

**getArmstrongSum**: This recursive function computes the sum of each digit raised to the power of the number of digits (num1). It extracts the last digit, raises it to the power of num1, and adds it to the recursive result of the remaining digits.

**num1**: This function counts the number of digits in the number. It keeps dividing the number by 10 until it becomes zero, it increments the count each time.

**main()**: The main function initializes a number (153, then 123), calculates the number of digits, and checks if the number equals the computed Armstrong sum. It prints the result accordingly.

-**Output**

****

****

-Time and Space complexity

**2. Prime Number**

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

Test Cases:

**Program:**

class ques2 {

static boolean isPrime(int num, int divisor) {

// if num is less than 2,not prime

if (num < 2)

return false;

// if divisor reaches num/2,it's prime

if (divisor \* divisor > num)

return true;

// If num is divisible by divisor,not prime

if (num % divisor == 0)

return false;

// check the next divisor

return isPrime(num, divisor + 1);

}

public static void main(String[] args) {

int num1 = 29;

if (isPrime(num1, 2)) {

System.out.println(num1 + " is a Prime Number");

} else {

System.out.println(num1 + " is not a Prime Number");

}

int num2 = 15;

if (isPrime(num2, 2)) {

System.out.println(num2 + " is a Prime Number");

} else {

System.out.println(num2 + " is not a Prime Number");

}

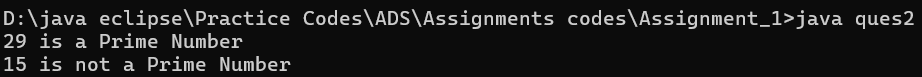
}

}

-Flow chart:

-Explanation:

-Output:



-Time and Space complexity

Input: 29

Output: true

Input: 15

Output: false

**3. Factorial**

**Problem: Write a Java program to compute the factorial of a given number.**

Program:

class ques3 { //Factorial

static int factorial(int num) {

if (num == 0) {

return 1;

}

return num \* factorial(num - 1);

}

public static void main(String[] args) {

int input1 = 5;

System.out.println("Factorial of " + input1 + " is: " + factorial(input1));

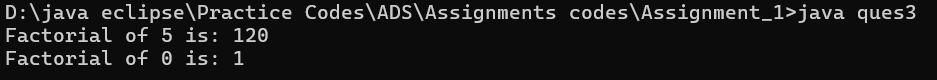
int input2 = 0;

System.out.println("Factorial of " + input2 + " is: " + factorial(input2));

}

}

Output:



Test Cases:

Input: 5

Output: 120

Input: 0

Output: 1

**4. Fibonacci Series**

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

Program:

class ques4{ //fibonacci

static int fib(int n){

if (n <= 1)

{

return n;

}

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

}

public static void main(String args[])

{

int num=6;

for(int i=0;i<=num;i++)

{

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

}

}

}

Output:



Test Cases:

Input: n = 5

Output: [0, 1, 1, 2, 3]

Input: n = 8

Output: [0, 1, 1, 2, 3, 5, 8, 13]

**5. Find GCD**

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

**Program:**

class ques5 { //GCD

static int gcd(int a, int b) {

if (b == 0) {

return a;

}

return gcd(b, a % b);

}

public static void main(String[] args) {

int a1 = 54, b1 = 24;

System.out.println("GCD of " + a1 + " and " + b1 + " is: " + gcd(a1, b1));

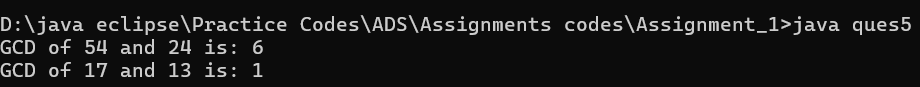
int a2 = 17, b2 = 13;

System.out.println("GCD of " + a2 + " and " + b2 + " is: " + gcd(a2, b2));

}

}

**Output:**



Test Cases:

Input: a = 54, b = 24

Output: 6

Input: a = 17, b = 13

Output: 1

**6. Find Square Root**

**Problem: Write a Java program to find the square root of a given number (using integer approximation).**

**Program:**

import java.math.RoundingMode;

import java.text.DecimalFormat;

class ques6 { //Square Root

public static void main(String args[])

{

DecimalFormat df = new DecimalFormat("#");

df.setRoundingMode(RoundingMode.CEILING);

double a = 27;

System.out.println(df.format(Math.sqrt(a)));

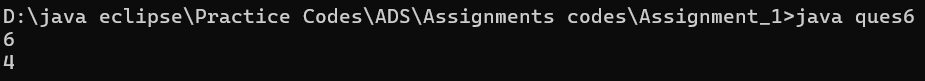
a = 16;

System.out.println(df.format(Math.sqrt(a)));

}

}

Output:



Test Cases:

Input: x = 16

Output: 4

Input: x = 27

Output: 5

**7. Find Repeated Characters in a String**

**Problem: Write a Java program to find all repeated characters in a string.**

**Program:**

import java.util.Scanner;

class ques7{ // Repeated characters

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

String str = sc.nextLine();

for(int i=0; i<str.length(); i++){

char thisChar = str.charAt(i);

for(int j=i+1; j<str.length(); j++){

if(thisChar == str.charAt(j)){

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

break;

}

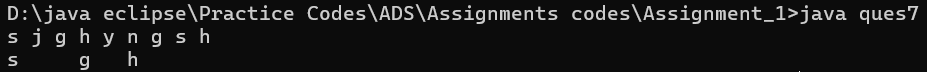
}

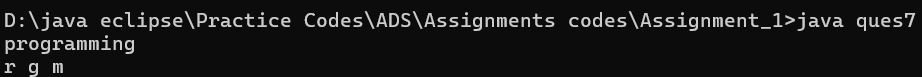
}

}

}

**Output:**

****

****

Test Cases:

Input: "programming"

Output: ['r', 'g', 'm']

Input: "hello"

Output: ['l']

**8. First Non-Repeated Character**

**Problem: Write a Java program to find the first non-repeated character in a string.**

**Program:**

import java.util.Scanner;

class ques8{

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

String str = sc.nextLine();

for(int i=0; i<str.length(); i++){

char thisChar = str.charAt(i);

boolean repeated = false;

for(int j=0; j<str.length(); j++){

if(i!=j && thisChar == str.charAt(j)){

repeated = true;

break;

}

}

if(!repeated){

System.out.println(thisChar);

break;

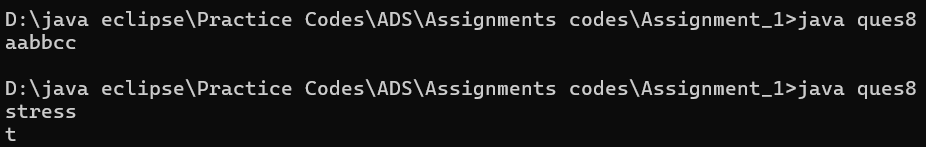
}

}

}

}

Output:



Test Cases:

Input: "stress"

Output: 't'

Input: "aabbcc"

Output: null

9**. Integer Palindrome**

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

Program:

class ques9 {

static boolean isPalindrome(int number) {

// fornegativevalue

if (number < 0) {

return false;

}

return checkPalindrome(number, 0);

}

static boolean checkPalindrome(int number, int reversed) {

// when number is 0, compare reversed with original

if (number == 0) {

return reversed == 0;

}

reversed = reversed \* 10 + number % 10;

return checkPalindrome(number / 10, reversed);

}

public static void main(String[] args) {

int input1 = 121;

System.out.println(input1 + " is a palindrome: " + isPalindrome(input1));

int input2 = -121;

System.out.println(input2 + " is a palindrome: " + isPalindrome(input2));

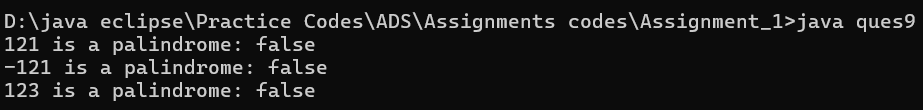
int input3 = 123;

System.out.println(input3 + " is a palindrome: " + isPalindrome(input3));

}

}

Output:



Test Cases:

Input: 121

Output: true

Input: -121

Output: false

**10. Leap Year**

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

**Program:**

import java.util.Scanner;

class LeapYear{

static boolean isLeapYear(int year){

if(year % 400 == 0 )

return true;

if(year % 100 == 0)

return false;

return year % 4 == 0;

}

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

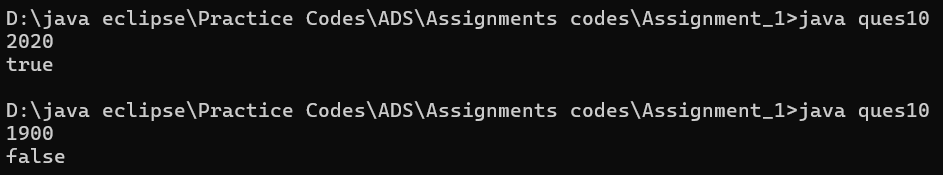
int year = sc.nextInt();

System.out.println(isLeapYear(year));

}

}

Output:



**Explanation :**

A year is a leap year if it is divisible by 400 or it is divisible by 4. But not divisible by 100.

**Time & Space Complexity :**

The time complexity for this program is O(1)

The space complexity of this program is O(1)

Test Cases:

Input: 2020

Output: true

Input: 1900

Output: false