**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

**import java.util.Scanner;**

**class ArmstrongNumberRecursion{**

**static int armSum(int number, int power)**

**{**

**// Base case: when number becomes 0, return 0**

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

**return 0;**

**}**

**// Get the last digit**

**int digit = number % 10;**

**// Recursive case: Add the nth power of the last digit to the result of remaining digits**

**return (int)Math.pow(digit,power)+armSum(number / 10 ,power);**

**}**

**// Helper method to calculate the number of digits in the number**

**static int count(int number){**

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

**return 0;**

**}**

**return 1+ count(number/10);**

**}**

**static boolean isArmNum(int number){**

**int numOfDigit = count(number);// Find the number of digits**

**return number == armSum(number , numOfDigit);// Check if the sum equals the original number**

**}**

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

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

**int n = sc.nextInt();**

**System.out.println("armstrong "+ isArmNum(n));**

**}**

**}**

2. Prime Number

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

Test Cases:

Input: 29

Output: true

Input: 15

Output: false

**import java.util.Scanner;**

**class Prime{**

**public static boolean isPrime(int num,int div){**

**if(num < 2)**

**return false;**

**if(div == 1)**

**return true;**

**if(num % div == 0)**

**return false;**

**return isPrime(num,div-1);**

**}**

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

**{**

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

**int num = sc.nextInt();**

**if(isPrime(num , num/2))**

**{**

**System.out.println(num +" true prime");**

**}**

**else{**

**System.out.println(num+" false not a prime");**

**}**

**}**

**}**

3. Factorial

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

Test Cases:

Input: 5

Output: 120

Input: 0

Output: 1

**import java.util.Scanner;**

**class facto{**

**public static int fact(int n){**

**if( n == 0)**

**return 1;**

**if(n <=1)**

**return 1;**

**else**

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

**}**

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

**System.out.println("enter number");**

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

**int n = sc.nextInt();**

**System.out.println("factorial : "+fact(n));**

**}**

**}**

4. Fibonacci Series

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

Test Cases:

Input: n = 5

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

Input: n = 8

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

**import java.util.Scanner;**

**public class FibonacciRe{**

**static int fibo(int n){**

**if(n <= 1)**

**return n;**

**else**

**return fibo(n-1)+fibo(n-2);**

**}**

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

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

**int num = sc.nextInt();**

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

**System.out.print(fibo(i)+" ");//0 1 1 2 3 5 8 13 21 34 55**

**}**

**}**

**}**

**//fib(3)=fib(2)+fib(1)**

**// =fib(1)+fib(0)+1**

**//f(n) = f(n-1) + f(n-2)**

5. Find GCD

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

Test Cases:

Input: a = 54, b = 24

Output: 6

Input: a = 17, b = 13

Output: 1

**//if(a>b)**

**//gcd(a%b,b)**

**//else(a,b%a)**

**import java.util.Scanner;**

**class GCD{**

**public static int findGcd(int a, int b)**

**{**

**if(b > a)//base condition**

**return findGcd(b,a);**

**if( b == 0)**

**return a;**

**else**

**return findGcd(b, a%b);**

**}**

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

**{**

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

**System.out.println("eneter any two number");**

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

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

**System.out.println("gcd : "+findGcd(a,b));**

**}**

**}**

6. Find Square Root

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

Test Cases:

Input: x = 16

Output: 4

Input: x = 27

Output: 5

**import java.util.Scanner;**

**class SquareRoot{**

**public static int Squareroot(int n )//n=4**

**{**

**//base condition**

**if( n ==0 || n==1)//false**

**return n;**

**// Starting from 1, try all numbers until**

**// i\*i is greater than or equal to x.**

**int i =1 , result =1;**

**while( result <= n){ // 1<= 4 -->true, true --> 4==4**

**i++; // i =2, i =3**

**result = i \* i; //result = 4 , 9**

**}**

**return i-1; //3-1 = 2**

**}**

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

**{**

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

**System.out.println("eneter any one number");**

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

**System.out.println("squareRoot : "+ Squareroot(a));// a= 4 , return "2"**

**}**

**}**

7. Find Repeated Characters in a String

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

Test Cases:

Input: "programming"

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

Input: "hello"

Output: ['l']

**class RepeatedChar{**

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

**{**

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

**System.out.println("Enter a string: ");**

**String str = new String(sc.nextLine());**

**char [] ch = str.toCharArray();**

**String temp = "";**

**for(int i =0; i<ch.length ; i++)**

**{**

**boolean isRepeated = false;**

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

**if(ch[i] == ch[j]){**

**isRepeated = true;**

**break;**

**}**

**}**

**if(isRepeated && str.indexOf(ch[i]) == i){**

**System.out.print(ch[i] + " ");**

**}**

**}**

**}**

**}**

8. First Non-Repeated Character

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

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.

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.

Test Cases:

Input: 2020

Output: true

Input: 1900

Output: false