**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

SOLUTION-**package** org.armstrongnumber.demo;

**import** java.util.Scanner;

**public** **class** ArmstrongNumberRecursion {

**static** **int** arm(**int** n)

{

**if**(n<10)

**return** n\*n\*n;

**return** (n%10)\*(n%10)\*(n%10)+*arm*(n/10);

}

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

**int** num, r;

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

System.***out***.println("Enter any Number");

num=sc.nextInt();

r = *arm*(num);

**if**(num==r)

System.***out***.println(num + " Is an Armstrong number");

**else**

System.***out***.println(num + " Is not an Armstrong number");

}

}

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

SOLUTION- **package** org.primenumber.demo;

**public** **class** PrimeNumberRecursion {

**static** **int** *i* = 2;

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

**if** (*isPrime*(29))

System.***out***.println("Number is Prime");

**else**

System.***out***.println("Number is Not Prime");

}

**public** **static** **boolean** isPrime(**int** n) {

**if** (*i* == n / 2 + 1)

**return** **true**;

**if** (n % *i* == 0)

**return** **false**;

*i*++;

**return** *isPrime*(n);

}

}

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

SOLUTION- **package** org.factorialnumber.demo;

**public** **class** FactorialNumberRecursion {

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

**if** (n <= 1)

**return** 1;

**else**

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

}

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

System.***out***.println(*fact*(5));// call for method

}

}

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]

SOLUTION-**package** org.fibonacci.demo;

**import** java.util.Scanner;

**public** **class** FibonacciSeriesRecursion {

**int** fib(**int** m) {

**if** (m == 0 || m == 1)

**return** m;

**else**

**return** fib(m - 1) + fib(m - 2);

}

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

FibonacciSeriesRecursion obj = **new** FibonacciSeriesRecursion();

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

System.***out***.println("Enter a limit");

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

System.***out***.println("Fibonacci series");

**for** (**int** i = 0; i < x; i++)

System.***out***.print(obj.fib(i) + ",");

}

}

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

SOLUTION- **package** org.gcd.demo;

**import** java.util.Scanner;

**public** **class** GcdRecursion {

**static** **int** gcd(**int** a, **int** b) {

**if** (a % b == 0)

**return** b;

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

}

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

**int** a, b;

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

System.***out***.println("Enter First Number");

a = sc.nextInt();

System.***out***.println("Enter Second Number");

b = sc.nextInt();

System.***out***.println("GCD : " + *gcd*(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

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']

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

SOLUTION-**package** org.leapyear.demo;

**import** java.util.Scanner;

**public** **class** LeapYearRecursion {

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

{

**if** (year <= 0)

**return** **false**;

}

**return** (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

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

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

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

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

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

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

} **else** {

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

}

}

}