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

**Subject: Algorithm and Data Structure**

**Assignment 1**

1. Armstrong Number

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

import java.util.Scanner;

import java.lang.Math;

class Armstrong{

static boolean isArmstrong(int n){

int temp,digits=0,last=0,sum=0;

temp = n;

while(temp>0){

temp = temp/10;

digits++;

}

temp=n;

while(temp>0){

last = temp%10;

sum+=(Math.pow(last,digits));

temp = temp/10;

}

if(n==sum){

return true;

}

else {

return false;

}

}

public static void main(String args[]){

int n1,n2;

Scanner sc = new Scanner(System.in);

n1 = sc.nextInt();

n2 = sc.nextInt();

if(isArmstrong(n1)){

System.out.println(n1 + "is Armstrong");

}else{

System.out.println(n1 + "is not Armstrong");

}

if(isArmstrong(n2)){

System.out.println(n2 + "is Armstrong");

}else{

System.out.println(n2 + "is not Armstrong");

}

}

}

output:

153

123

153 is Armstrong

123 is not Armstrong

Start

Input Number

Calculate sum of digits.

Compare original

True/False

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

2. Prime Number

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

class Prime{

static void isPrime(int n){

int i,m=0,flag=0;

m=n/2;

if(n==0 || n==1){

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

}

else{

for(i=2;i<=m;i++){

if(n%i==0){

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

flag=1;

break;

}

}

if(flag == 0){

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

}

}

}

public static void main(String args[]){

isPrime(29);

isPrime(15);

}

}

Start

Input number

Check prime

Return true/false

Output:

29 is prime number

15 is not prime number

**Time Complexity**: O(n)

**Space Complexity**: O(1)

3. Factorial

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

import java.util.Scanner;

class Factorial {

static int factorial(int n) {

if (n == 0 || n == 1) {

return 1;

} else {

return n \* factorial(n - 1);

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n1 = sc.nextInt();

int n2 = sc.nextInt();

int result1 = factorial(n1);

int result2 = factorial(n2);

System.out.println("Factorial of " + n1 + " is: " + result1);

System.out.println("Factorial of " + n2 + " is: " + result2);

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Factorial

5

0

Factorial of 5 is: 120

Factorial of 0 is: 1

Start

Input

number

Recursively calculate factorial

Return true/false

END

**Time Complexity**: O(n)

**Space Complexity**: O(n)

4. Fibonacci Series

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

class 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 n1 = 5;

int n2 = 8;

System.out.print("Fibonacci series up to " + n1 + ": ");

for (int i = 0; i < n1; i++) {

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

}

System.out.print("\nFibonacci series up to " + n2 + ": ");

for (int i = 0; i < n2; i++) {

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

}

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Fibonacci

Fibonacci series up to 5: 0 1 1 2 3

Fibonacci series up to 8: 0 1 1 2 3 5 8 13

**Time Complexity**:O(2^n)

**Space Complexity**: O(n)

Start

Input n value

Initialize I =0

Fibonacci(i)?IS I ==0? RETURN 0

Is I ==1 ? Return 1

Fabonacci(i)=(i

-

1)+(

i

-

2)

End

5. Find GCD

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

class GCD{

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

if(b==0){

return a;

}

return gcd(b,a%b);

}

public static void main(String args[]){

System.out.println(gcd(54,24));

System.out.println(gcd(17,13));

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java GCD

6

1

Input firt and second

Is b!=0?

Yes

Reminder =a%b

Start

No(end loop)

Update= a=b/

update b =rem

Is b!=0?

Yes

No(end loop)

End

**Time Complexity**: O(log⁡(n))

**Space Complexity**: O(log⁡(n))

6. Find Square Root

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

class Squareroot{

public static int sqrt(int n){

if(n==0||n==1){

return n;

}

int start =1,end=n/2,ans=0;

while(start<=end){

int mid=(start+end)/2;

if(mid\*mid==n)

return mid;

if(mid\*mid <n){

start = mid+1;

ans=mid;

}

else{

end = mid-1;

}

}

return ans;

}

public static void main(String args[]){

System.out.println(sqrt(16));

System.out.println(sqrt(27));

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Squareroot

4

5

Start

Input Number

Calculate Square root

Print answer

End

**Time Complexity**: O(log⁡(n))

**Space Complexity**: O(1)

7. Find Repeated Characters in a String

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

class Duplicate {

public static void duplicate(String str) {

int count;

char[] string = str.toCharArray();

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

count = 1;

if (string[i] != '\0') {

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

if (string[i] == string[j]) {

count++;

string[j] = '\0';

}

}

if (count > 1) {

System.out.println(string[i]);

}

}

}

}

public static void main(String[] args) {

duplicate("programming");

duplicate("hello");

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Duplicate

r

g

m

l

**Time Complexity**:O(n^2)

**Space Complexity**: O(n)

Start

Input String str

Initialize an empty hashmap

For each lop

End

8. First Non-Repeated Character

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

class Nonrepeat{

public static void Firstnonrepeat(String str){

boolean flag = true;

for(char i: str.toCharArray()){

if(str.indexOf(i)==str.lastIndexOf(i)){

System.out.println(i);

flag=false;

break;

}

}

if(flag==true){

System.out.println("null");

}

}

public static void main(String args[]){

Firstnonrepeat("stress");

Firstnonrepeat("aabbcc");

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Nonrepeat

t

null

Start

Input String str

Initialize an empty hashmap

For each lop

End

**Time Complexity**: O(n^2)

**Space Complexity**: O(1)

9. Integer Palindrome

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

class Palindrome {

public static void palindrome(int n) {

int r, sum = 0, temp;

temp = n;

if (n < 0) {

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

return;

}

while (n > 0) {

r = n % 10;

sum = (sum \* 10) + r;

n = n / 10;

}

if (temp == sum) {

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

} else {

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

}

}

public static void main(String args[]) {

palindrome(121);

palindrome(-121);

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Palindrome

121 is a palindrome

-121 is not a palindrome

**Time Complexity**: O(log⁡(n))

**Space Complexity**: O(1)

Start

Input number

Check if n is negative or 0.

Loop to Reverse the Number

Comparision

10. Leap Year

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

public class Leapyear {

public static void Leap(int yr){

if((yr % 4 == 0 && yr%100!=0) || (yr%400 ==0)) {

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

}

else {

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

}

}

public static void main(String[] args) {

Leap(2020);

Leap(1900);

}

}

C:\Users\saira\OneDrive\Documents\CDAC\ADS Module\Assignment1>java Leapyear

2020 is a leap year

1900 is not a leap year

Start

Input year

year% 400==0

,true

year %100 == 0

,false

End

**Time Complexity**: O(1)

**Space Complexity**: O(1)