1) Write a program to print your name.

Algorithm

Step-1) Start
Step-2) Read name String
Step-3) Print name
Step-4) End

Source Code

```
import java.util.Scanner;
class Main {
    public static void main (String args[]) {
        Scanner sc = new Scanner (System.in);
        System.out.print ("Enter name: ");
        String name = sc.nextLine();
        System.out.println ("Entered name:" + name);
    }
}
```

Output

```
Enter name: Sujal K B
Entered name: Sujal K B
```

2) Write a program to check whether the user inputted number is even or odd.

Algorithm

Step-1) Start
Step-2) Read n number ($n \in \mathbb{N}$)
Step-3) If n divide 2, gives remainder 0, go to step 4, otherwise go to step 5
Step-4) Print "Even", go to step 6
Step-5) Print "Odd", go to step 6
Step-6) End

Source Code

```
import java.util.Scanner;
class CheckEvenOdd {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a number:");
        int n = sc.nextInt();
        if(n%2==0){
            system.out.println("Even");
        } else {
            system.out.println("Odd");
        }
    }
}
```

Output (Testcases)

1) Enter a number: 26
Even

2) Enter a number: 27
Odd

3) Write a program to find largest and smallest number among three numbers.

Algorithm

Step-1) Start
Step-2) Declare a, b, c, ~~integers and~~ max, min integers.
Step-3) Read a, b, c
Step-4) Set max := a
Step-5) If b > max, go to step 6, otherwise go to step 7.
Step-6) Set max := b
Step-7) If c > max, go to step 8, otherwise go to step 9.
Step-8) Set max := c
Step-9) Set min := a
Step-10) If b < min, go to step 11, otherwise go to step 12.
Step-11) Set min := b
step-12) If c < min, go to step 13, otherwise go to step 14
Step-13) Set min := c
step-14) Print max, min
step-15) End

Source Code

```
import java.util.Scanner;
class LargeSmall {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter num1: ");
        int a = sc.nextInt();
        System.out.println("Enter num2:");
        int b= sc.nextInt();
        System.out.print("Enter num3:");
        int c = sc.nextInt();
        int max = a;
```

```
        if (b > max) {
            max = b;
        } if (c > max) {
            max = c;
        }
        int min = a;
        if (b < min) {
            min = b;
        } if (c < min) {
            min = c;
        }
        System.out.println("Largest: " + max + "\n Smallest: " +
                                                    min);
    }
}
```

Output

```
Enter num1 : 4
Enter num2 : 7
Enter num3 : 1
Largest : 7
Smallest: 1
```

4) Write a program to find GCD and LCM of two numbers.

Algorithm

step-1) Start
step-2) Declare function findGCD() with 2 numeric arguments from step 3 to step 5.
step-3) Read a, b integer from arguments.
step-4) If b equals 0, return a
step-5) return function findGCD(b, a%.b)
step-6) Declare n1, n2 ~~integers~~, gcd, lcm integers
step-7) Read n1, n2
step-8) Set gcd := findGCD(pass n1 and n2)
step-9) Set lcm := $\frac{n1 \times n2}{gcd}$
step-10) Print lcm, gcd
step-11) End

Source Code

```
import java.util.Scanner;
class Main{
   public static void main (String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter num1: ");
        int n1 = sc.nextInt();
        System.out.print("Enter num2:");
        int n2 = sc.nextInt();
        int gcd = findGCD(n1, n2);
        int lcm = (n1 * n2) / gcd;
        System.out.println("LCM:"+lcm+"\n GCD:"+gcd);
   }
   public static int findGCD(int a, int b){
        if(b==0)
            return a;
        return findGCD(b, a%.b);
   }
}
```

Output

```
Enter num1: 5
Enter num2: 15
LCM : 15
GCD : 5
```

5) Write a program to display all prime numbers between 1 to n.

Algorithm

Step-1) Start
Step-2) Declare n, low, i integers and f boolean variables.
Step-3) Read n
Step-4) Set low := 2
Step-5) Repeat steps 6 to step 12, until low < n:
Step-6) Set f := true
Step-7) Repeat step 8 to step 10; i = range(2, low):
step-8) If low/i gives remainder zero, go to step 9, otherwise step 7
Step-9) Set f := false
Step-10) Go to step 11
Step-11) If f equals true, PRINT low
Step-12) ~~Increment low by 1~~. Set low := low + 1

Source Code

```
import java.util.Scanner;
class Prime {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter n:");
        int n = sc.nextInt();
        int low = 2;
        System.out.println("Prime numbers from 1 to n: ");
        while (low<n){
            boolean f = true;
            for(int i=2; i<low; i++){
                if (low%i == 0){
                    f=false;
                    break;
                }
            }
            if(f==true){
                System.out.print(low+" ");
```

```
            }
            low++;
        }
    }
}
```

Output

```
Enter n: 10
Prime numbers from 1 to n:
2 3 5 7
```

6) Write a program to find the factorial of a user inputted value.

Algorithm

Step-1) Start
Step-2) Declare n, result integers.
Step-3) Read n
Step-4) Set result := 1
Step-5) Repeat step 6 to step 8 (until condition: n > 1),
Step-6) Set result := result * n
Step-7) Set n := n-1
Step-8) Print result
Step-9) End.

Source Code

```
import java.util.Scanner;
class Factorial {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt();
        int result = 1;
        while (n>1){
            result = result * n;
            n--;
        }
        System.out.println("Factorial: " + result);
    }
}
```

Output

```
Enter n: 5
Factorial: 120
```

7) Write a program to check whether the user inputted number is Armstrong or not.

Algorithm

~~1) Start~~

Step-1) Start
Step-2) Declare n, nCopy, d, result, lastDigit
Step-3) Read n
Step-4) Set nCopy := n
Step-5) Set d := 0
Step-6) Repeats steps 7 to 8 until (condition: n>0),
Step-7) Set n := n/10
Step-8) Set d := d+1
Step-9) Print d
Step-10) Set result := 0
Step-11) Set n := nCopy
Step-12) Repeat steps 13 to 15 (condition: nCopy >0),
Step-13) Set lastDigit := nCopy %.10
Step-14) Set result := result + (lastDigit)^d
Step-15) Set nCopy := nCopy/10
Step-16) Print result
Step-17) If n equals result, go to step 18 otherwise step 19
Step-18) Print "Armstrong". Go to step 20
Step-19) Print "Not Armstrong". Go to step 20
Step-20) End.

Source Code

```
import java.util.*;
class Armstrong {
    public static void main (String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n= sc.nextInt();
        int nCopy = n;
```

```
        int d=0;
        while (n>0){
            n=n/10;
            d++;
        }
        System.out.println("Digits: "+d);
        int result=0;
        int lastDigit;
        n=nCopy;
        while(nCopy>0){
            lastDigit=(nCopy%10);
            result+=Math.pow(lastDigit,d);
            nCopy=nCopy/10;
        }
        System.out.println("Result: "+result);
        if(n==result){
            System.out.println("Armstrong");
        }else{
            System.out.println("Not Armstrong");
        }
    }
}
```

Output (Testcases)

① Enter n: 153
Digits: 3
Result: 153
Armstrong

② Enter n: 154
Digits: 3
Result: 190
Not Armstrong

8) Write a program to print the fibonacci series till number n.

Algorithm

step-1) Start
step-2) Read n,i integers
step-3) Repeat steps 4 to 5 (condition i=0 to i<n)
step-4) Set result := fibonacci() of i.
step-5) Print result with ₿
step-6) function fibonacci from step 7 to step 9,
step-7) If n is 1, return 0
step-8) If n is 2, return 1
step-9) Otherwise fibonacci(n-1)+ fibonacci(n-2)
Step-10) End

Source Code

```
import java.util.Scanner;
class Fibonacci {
    public int findfibo(int n) {
        Fibonacci f= new Fibonacci();
        if(n==1) return 0;
        if(n==2) return 1;
        else return f.findfibo(n-1)+f.findfibo(n-2);
    }
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n= sc.nextInt();
        Fibonacci f = new Fibonacci();
        int i, result;
        System.out.print ("Fibonacci : ");
        for(i=1;i<=n;i++){
            result = f.findfibo(i);
            System.out.print(result+"₿");
    }
```

```
    }
}
```

Output

```
Enter n: 5
Fibonacci: 0 1 1 2 3
```

9) Write a program to find the square roots using sridharacharya formula.

Algorithm

Step-1) Start
Step-2) Declare a, b, c integers and D double variables
Step-3) Read a, b, c
Step-4) Set D := b^2-4ac // Discriminant
Step-5) If D>0, go to step 6, otherwise step 12
Step-6) Declare sqrtD, x1, x2 doubles variables
Step-7) Set sqrtD := $\sqrt{D}$
Step-8) Set x1 := $\frac{-b+sqrtD}{2a}$
Step-9) Set x2 := $\frac{-b-sqrtD}{2a}$
Step-10) Print "Roots are real and distinctive."
Step-11) Print x1, x2, go to step 23
Step-12) If D=0, go to step 13, otherwise step 17
Step-13) Declare x integer variable
Step-14) Read x → set x := $\frac{-b}{2a}$
Step-15) Print "Roots are real and equal"
Step-16) Print x, go to step 23
Step-17) If D<0, go to step 18, otherwise step 23.
Step-18) Declare realPart, imaginaryPart double variables.
Step-19) Set realPart := $\frac{-b}{2a}$
Step-20) Set imaginaryPart := $\frac{\sqrt{-D}}{2a}$
Step-21) Print "Roots are complex and imaginary"
Step-22) Print x1 = realPart + imaginaryPart, x2 = realPart + imaginaryPart
Step-23) End

Source Code

```
import java.util.*;
class Sridharacharya {
```

```
public static void main(String args[]){
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a: ");
    int a = sc.nextInt();
    System.out.print("Enter b: ");
    int b = sc.nextInt();
    System.out.print("Enter c: ");
    int c = sc.nextInt();
    double D = b*b - 4*a*c;
    if (D>0){
        double sqrtD = Math.sqrt(D);
        double x1 = (-b+sqrtD) / (2*a);
        double x2 = (-b-sqrtD) / (2*a);
        System.out.println("Roots are real and distinct!");
        System.out.println("x1="+x1+"\nx2="+x2);
    } else if (D==0){
        double x= -b / (2.0*a);
        System.out.println("Roots are real and equal:");
        System.out.println("x ="+x);
    } else if (D<0){
        double realPart = -b/(2.0*a);
        double imagPart = Math.sqrt(-D)/(2.0*a);
        System.out.println("Roots are complex and imaginary");
        System.out.println("x1="+realPart+"+"+imagPart+"i");
        System.out.println("x2="+realPart+"-"+imagPart+"i");
    }
}
}
```

output

```
Enter a: 1
Enter b: -5
Enter c: 6
Roots are real and distinctive:
x1=3.0
x2=2.0
```

10) Write a program to print the sum of this series-

$$1 - \frac{1}{2!} + \frac{1}{3!} - \frac{1}{4!} + \ldots + \frac{1}{n!}$$

Algorithm

Step-1) Start
Step-2) Declare n, result, sign, i
Step-3) Read n
Step-4) Set result := 0
Step-5) Set sign := 1
Step-6) Repeat step 7 to step 8 (i = 1 to i <= n)
Step-7) Set result := sign x $\left(\frac{1}{i!}\right)$
Step-8) Set sign := sign x (-1)
Step-9) Print result.
Step-10) End

Source Code

```
import java.util.Scanner;
class Series1 {
    public int facto(int x){
        int result = 1;
        while (x>1){
            result *= x;
            x--;
        }
        return result;
    }
    public static void main(String args[]) {
        Series1 s = new Series1();
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt();
        double result = 0;
        int sign = 1;
        for(int i=1; i<=n; i++){
```

```
            result += sign * (1.0/S.facto(i));
            sign *= -1;
        }
        System.out.println("Result: "+ result);
    }
}
```

Output

```
Enter n: 3
Result: 0.6666666666666666
```

11.) Write a program to find the sum of this series -

$$1+\frac{1}{2!}-\frac{1}{3!}+\frac{1}{4!}-\ldots+\frac{1}{n!}$$

Algorithm

Step-1) Start
Step-2) Declare n, result, sign, i
Step-3) Read n
Step-4) Set result := 1
Step-5) Set sign := 1
Step-6) Repeat step 7 to step 8 (i = 2 to i ≤ n)
Step-7) Set result := result + sign / i!
Step-8) set sign := sign * (-1)
Step-9) Print result
Step-10) End

Source Code

```
import java.util.*;
class Series2 {
    public int facto (int a){
        int result = 1;
        while (a>1){
            result *= a;
            a--;
        }
        return result;
    }
    public static void main (String args[]){
        Series2 s = new Series2();
        Scanner sc = new Scanner (System.in);
        System.out.print ("Enter n: ");
        int n = sc.nextInt();
        double result = 1;
        int sign = 1;
        for (int i=2; i<=n; i++){
            result += sign * (1.0/s.facto(i));
            sign *= -1;
        }
        System.out.println("Result: "+result);
    }
}
```

Output

```
Enter n: 3
Result: 1.3333333333
```

12) Write a program to print the diamond pattern.

Algorithm

Step-1) Start
Step-2) Declare n,i,j integer variables
Step-3) Read n
Step-4) Repeat step 5 to step 9 (i=1 to i≤n)
Step-5) Repeat step 6 (j=1 to j<n)
Step-6) Print " "
Step-7) Repeat step 8 (j=1 to j≤2i-1)
Step-8) Print "*"
Step-9) Print new line
Step-10) Repeat step 11 to step 15 (i=n-1 to i≥1)
Step-11) Repeat step 12 (j=n to j>i)
Step-12) Print " "
Step-13) Repeat step 14 (j=1 to j≤2i-1)
Step-14) Print *
Step-15) Print new line
Step-16) End

Source Code

```
import java.util.Scanner;
class Pattern1{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n = sc.nextInt();
        for(int i=1 ; i<=n ; i++){          // Upper Half
            for(int j = i ; j<n ; j++){
                System.out.print(" ");
            }
            for(int j = 1 ; j <= 2*i-1 ; j++){
                System.out.print("*");
            }
            System.out.println();
        }
```

```
        for(int i=n-1; i>=1; i--){          // Lower Half
            for(int j=n; j>1; j--){
                System.out.print(" ");
            }
            for(int j=1; j<=2*i-1; j++){
                System.out.print("*");
            }
            System.out.println();
        }
    }
}
```

Output

```
Enter n: 3
    *
  * * *
* * * * *
  * * *
    *
```

13) Write a program to print X pattern.

Algorithm

step-1) Start
step-2) Declare n, i, j integer variables
step-3) Read n
step-4) If n is odd, go to step 12
step-5) Repeat step 6 to step 11. (i=0 till i<n)
step-6) Repeat step 7 to step 10. (j=0 till j<n)
step-8) If j=i or j=n-1-i, then go to step 9, otherwise step 10.
step-9) Print "*"
step-10) Print " "
step-11) Print new line
step-12) End

Source Code

```
import java.util.Scanner;
class Pattern1{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter n: ");
        int n= sc.nextInt();
        if(n%2==0){
            System.out.println("Please enter an odd number.");
            return;
        }
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                if(j==i||j==n-1-i)
                    System.out.print("*");
                else
                    System.out.print(" ");
            }
            System.out.println()
        }
    }
}
```

P.T.O.

Output

① Enter n: 3

② Enter n: 4
Please enter an odd number.

14) Write a program to find the multiplication of a user inputted matrices.

Algorithm

Step-1) Start
Step-2) Declare m, n, p, q, i, j, k integers and A[][], B[][], result[][] integer matrices.
Step-3) Read m, n, p, q
Step-4) Read A matrix of order m x n
Step-5) Read B matrix of order p x q
Step-6) If n==p, go to step 14.
Step-7) Repeat step 8 to step 13, (condition: i=0 to i<m1)
Step-8) Repeat step 9 to step 12, (condition: j=0 to j<n2)
Step-9) Repeat step 10 to step 11, (condition: k=0 to k<n1)
Step-10) Set result[i][j]:= result[i][j] + (A[i][k] X B[k][j])
Step-11) Set k:= k+1
Step-12) Set j:= j+1
Step-13) Set i:= i+1
Step-14) End

Source Code

```
import java.util.Scanner;
class MatrixMultiplication {
    public static void main (String args[]) {
        Scanner sc = new Scanner(System.in);
        System.out.print ("Enter rows of first matrix:");
        int m1 = sc.nextInt();
        System.out.print("Enter columns of first matrix:");
        int n1 = sc.nextInt();
        System.out.print("Enter rows of second matrix:");
        int m2 = sc.nextInt();
        System.out.print("Enter columns of second matrix:");
```

```
int n2 = sc.nextInt();
if(n1 != m2) {
    System.out.println("Matrix multiplication not
                        possible.");
    return;
}
int[][] A = new int[m1][n1];
int[][] B = new int[m2][n2];
int[][] result = new int[m1][n2];
System.out.println("Enter elements of first matrix:");
for(int i=0; i<m1; i++)
    for(int j=0; j<n1; j++)
        A[i][j] = sc.nextInt();
System.out.println("Enter elements of second matrix:");
for(int i=0; i<m2; i++)
    for(int j=0; j<n2; j++)
        B[i][j] = sc.nextInt();
//Matrix Multiplication
for(int i=0; i<m1; i++) {
    for(int j=0; j<n2; j++) {
        for(int k=0; k<n1; k++) {
            result[i][j] += A[i][k] * B[k][j];
        }
    }
}
//Result
System.out.println("Resultant matrix:");
for(int i=0; i<m1; i++) {
    for(int j=0; j<n2; j++) {
        System.out.print(result[i][j] + "\t");
    }
    System.out.println();
}
}
}
```

Output

```
Enter rows of first matrix: 2
Enter columns of first matrix: 2
Enter rows of first matrix: 2
Enter columns of second matrix: 2
Enter elements of first matrix:
1   2
3   4
Enter elements of second matrix:
1   0
0   1
Resultant matrix:
1   2
3   4
```

15) Write a program to find the sum of even numbers of a user inputted 1D array.

Algorithm

Step-1) Start
Step-2) Read n (where, n = size of 1D arrays)
Step-3) Read A array of size n.
Step-4) Declare sum, i
Step-5) Set sum := 0
Step-6) Repeat step 7 to step 9, (condition i=0 to i<n)
Step-7) If A's i^{th} element is even, go to step 8, otherwise go to step 9.
Step-8) Set sum := sum + A[i]
Step-9) Set i := i+1
Step-10) Print sum
Step-11) End

Source Code

```
import java.util.Scanner;
class EvenSumArray {
    public static void main (String args[]) {
        Scanner sc = new Scanner (System.in);
        System.out.print ("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] A = new int[n];
        int sum = 0;
        System.out.println ("Enter "+n+" elements:");
        for (int i=0; i<n; i++) {
            arr[i] = sc.nextInt();
            if (arr[i]%2 == 0) {
                sum += arr[i];
            }
        }
        System.out.println ("Sum of even numbers:" + sum);
    }
}
```

Output

```
Enter the size of the array: 5
Enter 5 elements:
1 2 3 4 5
Sum of even numbers: 6
```

16) Write a program to find the row sum and column sum in two respective 1D arrays.

Algorithm

Step-1) Start
Step-2) Declare rows, cols, matrix[rows][cols], rowSum[rows], colSum[cols], i, j
Step-3) Read rows, cols
Step-4) Repeat step 5 to step 10, (condition i=0 to i<rows)
Step-5) Repeat step 6 to step 9, (condition j=0 to j<cols)
Step-6) Read matrix[i][j]
Step-7) Set rowSum[i] := rowSum[i] + matrix[i][j];
Step-8) Set colSum[j] := colSum[j] + matrix[i][j]
Step-9) Set j := j+1
Step-10) Set i := i+1
Step-11) Print rowSum, colSum
Step-12) End

Source Code

```
import java.util.Scanner;
class RowColSum{
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter number of rows:");
        int rows = sc.nextInt();
        S.out("Enter no. of cols:");
        int cols = sc.nextInt();
        int[][] matrix = new int[rows][cols];
        int[] rowSum = new int[rows];
        int[] colSum = new int[cols];
        System.out.println("Enter matrix elements:");
        for(int i=0; i<rows; i++){
            for(int j=0; j<cols; j++){
                matrix[i][j] = sc.nextInt();
                rowSum[i] += matrix[i][j];
```

```
                colSum[j] += matrix[i][j];
            }
        }
        System.out.println("Row sum array: ");
        for(int i=0; i<rows; i++){
            System.out.print(rowSum[i]+"\t");
        }
        System.out.println("\n Column sum array: ");
        for(int j=0; j<cols; j++){
            System.out.print(colSum[j]+"\t");
        }
    }
}
```

Output

```
Enter number of rows: 2
Enter no. of cols: 2
Enter matrix elements:
1   2
3   4
Row sum array:
3   7
Column sum array:
4   6
```