1. **Hollow square**

**Pseudo code:**

Start

Input n (size of the square)

For i from 1 to n: // Loop through rows

For j from 1 to n: // Loop through columns

If i is 1 or i is n or j is 1 or j is n:

Print "\*"

Else:

Print " "

End For

Print new line (Move to the next row)

End For

End

**Java code:**

import java.util.Scanner;

public class HollowSquare {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++)

System.out.print((i == 1 || i == n || j == 1 || j == n) ? "\*" : " ");

System.out.println();

}

}

}

1. **Print odd numbers after 4 odd numbers**

**Pseudo code:**

Start

Initialize count = 0

Initialize current = 1

While count < 4:

If current is odd:

Increment count by 1

Increment current by 2 // Move to the next odd number

Print "Odd numbers after the first 4 odd numbers:"

While true:

Print current

Increment current by 2 // Print the next odd number

End

**Java code:**

import java.util.Scanner;

public class OddNumbersAfterFour {

public static void main(String[] args) {

int count = 0, current = 1;

// Skip the first 4 odd numbers

while (count < 4) {

count++;

current += 2;

}

System.out.println("Odd numbers after the first 4 odd numbers:");

// Print odd numbers indefinitely

while (true) {

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

current += 2;

}

}

}

1. **Factorial**

**Pseudo code:**

Start

Input n (the number to calculate the factorial)

Set result = 1

For i from 1 to n:

result = result \* i

End For

Output result (the factorial of n)

End

**Java code:**

import java.util.Scanner;

public class Factorial {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

int fact = 1;

for (int i = 1; i <= n; i++)

fact \*= i;

System.out.println(fact);

}

}

**4.fibonacci**

**Pseudo code:**

Start

Input n (the number of terms in the Fibonacci sequence)

Initialize a = 0, b = 1

Print a

Print b

For i from 3 to n:

next = a + b

Print next

a = b

b = next

End For

End

**Java code:**

import java.util.Scanner;

public class Fibonacci {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

int a = 0, b = 1;

System.out.print(a + " " + b);

for (int i = 3; i <= n; i++) {

int next = a + b;

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

a = b;

b = next;

}

}

}

**5.pyramid pattern**

**Pseudo code:**

Start

Input n (number of rows for the pyramid)

For i from 1 to n: // Loop through rows

Print (n - i) spaces // To align the stars in a pyramid shape

Print (2 \* i - 1) stars // Print odd number of stars in each row

Move to the next line

End For

End

**Java code:**

import java.util.Scanner;

public class PyramidPattern {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

for (int i = 1; i <= n; i++) {

System.out.print(" ".repeat(n - i)); // Print spaces

System.out.println("\*".repeat(2 \* i - 1)); // Print stars

}

}

}

**6.inverted pyramid**

**Pseudo code:**

Start

Input n (number of rows for the pyramid)

For i from n to 1: // Loop through rows

Print (n - i) spaces // To align the stars in an inverted pyramid shape

Print (2 \* i - 1) stars // Print odd number of stars in each row

Move to the next line

End For

End

**Java code:**

import java.util.Scanner;

public class InvertedPyramid {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

for (int i = n; i >= 1; i--) {

System.out.print(" ".repeat(n - i)); // Print spaces

System.out.println("\*".repeat(2 \* i - 1)); // Print stars

}

}

}

**7.factors of n**

**Pseudo code:**

Start

Input n (the number to find factors of)

For i from 1 to n:

If n % i == 0: // If i divides n with no remainder

Print i

End If

End For

End

**Java code:**

import java.util.Scanner;

public class Factors {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

for (int i = 1; i <= n; i++) {

if (n % i == 0) {

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

}

}

}

}

**8. Armstrong number**

**Pseudo code:**

Start

Input n (the number to check)

Initialize sum = 0

Initialize temp = n

Find the number of digits in n (let's call it d)

While temp is greater than 0:

Extract the last digit of temp (digit = temp % 10)

Add digit^d to sum (sum = sum + digit^d)

Remove the last digit from temp (temp = temp / 10)

End While

If sum == n:

Print "Armstrong number"

Else:

Print "Not an Armstrong number"

End

**Java code:**

import java.util.Scanner;

public class Armstrong {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

int sum = 0, temp = n, digits = String.valueOf(n).length();

while (temp > 0) {

int digit = temp % 10;

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

temp /= 10;

}

System.out.println(sum == n ? "Armstrong number" : "Not an Armstrong number");

}

}

**9. matrix multiplication**

**Pseudo code:**Start

Input matrix A of size m x n

Input matrix B of size n x p

Initialize matrix C of size m x p with 0s

For i from 0 to m-1: // Iterate over rows of A

For j from 0 to p-1: // Iterate over columns of B

For k from 0 to n-1: // Iterate for multiplication

C[i][j] = C[i][j] + A[i][k] \* B[k][j]

End For

End For

End For

Print matrix C

End

**Java code:**

import java.util.Scanner;

public class MatrixMultiplication {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input dimensions

System.out.print("Enter rows and columns for Matrix A: ");

int m = scanner.nextInt();

int n = scanner.nextInt();

int[][] A = new int[m][n];

System.out.println("Enter elements of Matrix A:");

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

for (int j = 0; j < n; j++)

A[i][j] = scanner.nextInt();

System.out.print("Enter columns for Matrix B: ");

int p = scanner.nextInt();

int[][] B = new int[n][p];

System.out.println("Enter elements of Matrix B:");

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

for (int j = 0; j < p; j++)

B[i][j] = scanner.nextInt();

// Initialize result matrix C

int[][] C = new int[m][p];

// Matrix multiplication

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

for (int j = 0; j < p; j++)

for (int k = 0; k < n; k++)

C[i][j] += A[i][k] \* B[k][j];

// Print result matrix C

System.out.println("Resultant Matrix C:");

for (int[] row : C) {

for (int val : row)

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

System.out.println();

}

}

}

**10.square, cube**

**Pseudo code:**

Start

Input n (the number)

square = n \* n

cube = n \* n \* n

Print "Square: " + square

Print "Cube: " + cube

End

**Java code:**

import java.util.Scanner;

public class SquareAndCube {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

int square = n \* n;

int cube = n \* n \* n;

System.out.println("Square: " + square);

System.out.println("Cube: " + cube);

}

}

**11.right triangle patter(number)**

**Pseudo code:**

Start

Input n (number of rows for the triangle)

For i from 1 to n:

For j from 1 to i:

Print j

End For

Print newline

End For

End

**Java code:**

import java.util.Scanner;

public class RightAngleTriangle {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

for (int i = 1; i <= n; i++) {

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

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

}

System.out.println(); // Move to the next line after each row

}

}

}

**12.right triangle \* pattern**

**Pseudo code:**

Start

Input n (number of rows for the triangle)

For i from 1 to n:

For j from 1 to i:

Print "\*"

End For

Print newline

End For

End

**Java code:**

import java.util.Scanner;

public class RightAngleTriangleStar {

public static void main(String[] args) {

int n = new Scanner(System.in).nextInt();

for (int i = 1; i <= n; i++) {

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

System.out.print("\* ");

}

System.out.println(); // Move to the next line after each row

}

}

}

**13.perfect number**

**Pseudo code:**

Start

Input n (the number to check)

Initialize sum = 0

For i from 1 to n-1:

If n % i == 0: // If i is a divisor of n

sum = sum + i

If sum == n:

Print "n is a perfect number"

Else:

Print "n is not a perfect number"

End

**Java code:**

import java.util.Scanner;

public class PerfectNumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

int sum = 0;

// Calculate the sum of divisors

for (int i = 1; i < n; i++) {

if (n % i == 0) {

sum += i;

}

}

// Check if the sum of divisors equals the number

if (sum == n) {

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

} else {

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

}

}

}

**14.strong number**

**Pseudo code:**

Start

Input n (the number to check)

Initialize sum = 0

Set originalNumber = n

While n > 0:

Extract last digit (digit = n % 10)

Find factorial of digit

Add factorial to sum

Remove the last digit (n = n / 10)

If sum == originalNumber:

Print "n is a strong number"

Else:

Print "n is not a strong number"

End

**Java code:**

import java.util.Scanner;

public class StrongNumber {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

int originalNumber = n;

int sum = 0;

// Function to calculate factorial

int factorial(int num) {

int fact = 1;

for (int i = 1; i <= num; i++) {

fact \*= i;

}

return fact;

}

// Calculate sum of factorials of digits

while (n > 0) {

int digit = n % 10;

sum += factorial(digit);

n /= 10;

}

// Check if the sum of factorials equals the original number

if (sum == originalNumber) {

System.out.println(originalNumber + " is a strong number.");

} else {

System.out.println(originalNumber + " is not a strong number.");

}

}

}

**15.square pattern**

**Pseudo code:**

Start

Input n (size of the square)

For i from 1 to n:

For j from 1 to n:

Print "\*"

End For

Print newline

End For

End

**Java code:**

import java.util.Scanner;

public class SquarePattern {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

for (int i = 1; i <= n; i++) {

for (int j = 1; j <= n; j++) {

System.out.print("\* ");

}

System.out.println(); // Move to the next line after each row

}

}

}