**Assignment Number:2**

**Solution 7:**

#include <iostream>

using namespace std;

unsigned long long factorial(int n) {

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

return 1;

else

return n \* factorial(n - 1);

}

int main() {

int num;

cout << "Enter a number to find its factorial: ";

cin >> num;

cout << "Factorial of " << num << " is: " << factorial(num) << endl;

return 0;

}

**Solution 8:**

#include <iostream>

using namespace std;

void multiplicationTable(int num) {

cout << "Multiplication Table of " << num << ":" << endl;

for (int i = 1; i <= 10; ++i) {

cout << num << " \* " << i << " = " << (num \* i) << endl;

}

}

int main() {

int number;

cout << "Enter a number to create its multiplication table: ";

cin >> number;

multiplicationTable(number);

return 0;

}

**Solution 9:**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter the number of terms in the Fibonacci series: ";

cin >> n;

int first = 0, second = 1, next;

cout << "Fibonacci Series:" << endl;

cout << first << " " << second << " ";

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

next = first + second;

cout << next << " ";

first = second;

second = next;

}

return 0;

}

**Solution 10:**

#include <iostream>

using namespace std;

int findHCF(int num1, int num2) {

while (num2 != 0) {

int temp = num2;

num2 = num1 % num2;

num1 = temp;

}

return num1;

}

int main() {

int number1, number2;

cout << "Enter the first number: ";

cin >> number1;

cout << "Enter the second number: ";

cin >> number2;

int hcf = findHCF(number1, number2);

cout << "The HCF of " << number1 << " and " << number2 << " is: " << hcf << endl;

return 0;

}

**Solution 11:**

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

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

if (i % 2 == 0) {

cout << i << " is even." << endl;

} else {

cout << i << " is odd." << endl;

}

}

return 0;

}

**Solution 13:**

#include <iostream>

using namespace std;

int main() {

const int size = 5; // Size of the array

int arr[size];

cout << "Enter " << size << " elements:" << endl;

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

cin >> arr[i];

}

int maxElement = arr[0];

int minElement = arr[0];

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

if (arr[i] > maxElement) {

maxElement = arr[i];

}

if (arr[i] < minElement) {

minElement = arr[i];

}

}

cout << "Maximum element in the array: " << maxElement << endl;

cout << "Minimum element in the array: " << minElement << endl;

return 0;

}

**Solution 14:**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

double a, b, c;

double root1, root2, imaginaryPart;

cout << "Enter coefficients a, b, and c: ";

cin >> a >> b >> c;

double discriminant = b \* b - 4 \* a \* c;

if (discriminant > 0) {

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

cout << "Roots are real and distinct." << endl;

cout << "Root 1 = " << root1 << endl;

cout << "Root 2 = " << root2 << endl;

} else if (discriminant == 0) {

root1 = root2 = -b / (2 \* a);

cout << "Roots are real and equal." << endl;

cout << "Root 1 = Root 2 = " << root1 << endl;

} else {

realPart = -b / (2 \* a);

imaginaryPart = sqrt(-discriminant) / (2 \* a);

cout << "Roots are complex and imaginary." << endl;

cout << "Root 1 = " << realPart << " + " << imaginaryPart << "i" << endl;

cout << "Root 2 = " << realPart << " - " << imaginaryPart << "i" << endl;

}

return 0;

}

**Solution 15:**

#include <iostream>

using namespace std;

int main() {

int num1, num2, num3;

cout << "Enter three numbers: ";

cin >> num1 >> num2 >> num3;

int max = num1;

if (num2 > max) {

max = num2;

}

if (num3 > max) {

max = num3;

}

cout << "The greatest number among " << num1 << ", " << num2 << ", and " << num3 << " is: " << max << endl;

return 0;

}

**Solution 16:**

#include <iostream>

using namespace std;

const int MAX\_ROWS = 100;

const int MAX\_COLS = 100;

int main() {

int matrix[MAX\_ROWS][MAX\_COLS];

int rows, cols;

cout << "Enter the number of rows and columns of the matrix: ";

cin >> rows >> cols;

cout << "Enter the elements of the matrix:" << endl;

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

for (int j = 0; j < cols; ++j) {

cin >> matrix[i][j];

}

}

cout << "Sum of each row:" << endl;

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

int rowSum = 0;

for (int j = 0; j < cols; ++j) {

rowSum += matrix[i][j];

}

cout << "Row " << i + 1 << ": " << rowSum << endl;

}

return 0;

}

**Solution 17:**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

int main() {

int matrix[MAX\_SIZE][MAX\_SIZE];

int rows, cols;

cout << "Enter the number of rows and columns of the matrix: ";

cin >> rows >> cols;

cout << "Enter the elements of the matrix:" << endl;

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

for (int j = 0; j < cols; ++j) {

cin >> matrix[i][j];

}

}

bool isDiagonal = true;

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

for (int j = 0; j < cols; ++j) {

// If an off-diagonal element is non-zero, it's not a diagonal matrix

if (i != j && matrix[i][j] != 0) {

isDiagonal = false;

break;

}

}

if (!isDiagonal) {

break;

}

}

if (isDiagonal) {

cout << "The given matrix is a diagonal matrix." << endl;

} else {

cout << "The given matrix is not a diagonal matrix." << endl;

}

return 0;

}

**Solution 18:**

#include <iostream>

using namespace std;

int main() {

int rows;

cout << "Enter the number of rows for the pattern: ";

cin >> rows;

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

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

cout << j << " ";

}

cout << endl;

}

return 0;

}

**Solution 19:**

#include <iostream>

using namespace std;

int main() {

int rows;

char specialChar;

cout << "Enter the number of rows for the pattern: ";

cin >> rows;

cout << "Enter the special character to use: ";

cin >> specialChar;

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

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

cout << specialChar << " ";

}

cout << endl;

}

return 0;

}

**Solution 20:**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

int main() {

int rows, cols;

cout << "Enter the number of rows and columns for the matrices: ";

cin >> rows >> cols;

int matrix1[MAX\_SIZE][MAX\_SIZE];

cout << "Enter the elements of the first matrix:" << endl;

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

for (int j = 0; j < cols; ++j) {

cin >> matrix1[i][j];

}

}

int matrix2[MAX\_SIZE][MAX\_SIZE];

cout << "Enter the elements of the second matrix:" << endl;

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

for (int j = 0; j < cols; ++j) {

cin >> matrix2[i][j];

}

}

int result[MAX\_SIZE][MAX\_SIZE];

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

for (int j = 0; j < cols; ++j) {

result[i][j] = matrix1[i][j] + matrix2[i][j];

}

}

cout << "Resultant matrix after addition:" << endl;

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

for (int j = 0; j < cols; ++j) {

cout << result[i][j] << " ";

}

cout << endl;

}

return 0;

}

**Solution 21:**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

int main() {

int size, index, newValue;

cout << "Enter the size of the array: ";

cin >> size;

int arr[MAX\_SIZE];

cout << "Enter the elements of the array:" << endl;

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

cin >> arr[i];

}

cout << "Enter the index of the element to update: ";

cin >> index;

if (index < 0 || index >= size) {

cout << "Invalid index! Program exiting." << endl;

return 1;

}

cout << "Enter the new value for the element: ";

cin >> newValue;

arr[index] = newValue;

cout << "Updated array:" << endl;

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

cout << arr[i] << " ";

}

cout << endl;

return 0;

}

**Solution 22:**

#include <iostream>

#include <cstring>

using namespace std;

const int MAX\_LENGTH = 100;

int main() {

char str1[MAX\_LENGTH], str2[MAX\_LENGTH];

cout << "Enter the first string: ";

cin.getline(str1, MAX\_LENGTH);

cout << "Enter the second string: ";

cin.getline(str2, MAX\_LENGTH);

strcat(str1, str2);

cout << "Concatenated string: " << str1 << endl;

return 0;

}

**Solution 23:**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char str[100];

cout << "Enter a string: ";

cin.getline(str, 100);

int size = strlen(str);

cout << "Size of the string: " << size << endl;

return 0;

}

**Solution 24:**

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

bool isDiagonalMatrix(int arr[MAX\_SIZE][MAX\_SIZE], int n) {

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

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

if (i != j && arr[i][j] != 0) {

return false;

}

}

}

return true;

}

int main() {

int n;

cout << "Enter the size of the square matrix: ";

cin >> n;

int matrix[MAX\_SIZE][MAX\_SIZE];

cout << "Enter the elements of the matrix:" << endl;

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

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

cin >> matrix[i][j];

}

}

if (isDiagonalMatrix(matrix, n)) {

cout << "The matrix is a diagonal matrix." << endl;

} else {

cout << "The matrix is not a diagonal matrix." << endl;

}

return 0;

}

**Solution 25:**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char str1[100] = "Hello, ";

char str2[] = "world!";

strcat(str1, str2);

cout << "Concatenated string: " << str1 << endl;

return 0;

}

**Solution 26:**

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char source[] = "Hello, world!";

char destination[100];

strcpy(destination, source);

cout << "Copied string: " << destination << endl;

return 0;

}

**Solution 27:**

#include <iostream>

using namespace std;

bool isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i \* i <= num; i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

int main() {

int limit;

cout << "Enter the limit: ";

cin >> limit;

cout << "Prime numbers up to " << limit << ": ";

for (int i = 2; i <= limit; i++) {

if (isPrime(i)) {

cout << i << " ";

}

}

cout << endl;

return 0;

}

**Solution 28:**

#include <iostream>

using namespace std;

unsigned long long factorialWithoutRecursion(int n) {

unsigned long long fact = 1;

while (n > 1) {

fact \*= n;

n--;

}

return fact;

}

int main() {

int number;

cout << "Enter a positive integer: ";

cin >> number;

if (number < 0) {

cout << "Factorial is not defined for negative numbers." << endl;

} else {

unsigned long long result = factorialWithoutRecursion(number);

cout << "Factorial of " << number << " is: " << result << endl;

}

return 0;

}

**Solution 29:**

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number to generate its multiplication table: ";

cin >> number;

int i = 1;

cout << "Multiplication table of " << number << ":" << endl;

while (i <= 10) {

cout << number << " \* " << i << " = " << (number \* i) << endl;

i++;

}

return 0;

}

**Solution 30:**

#include <iostream>

using namespace std;

int main() {

int n, firstTerm = 0, secondTerm = 1, nextTerm;

cout << "Enter the number of terms: ";

cin >> n;

cout << "Fibonacci Series:" << endl;

int i = 1;

while (i <= n) {

if(i == 1) {

cout << firstTerm << ", ";

i++;

continue;

}

if(i == 2) {

cout << secondTerm << ", ";

i++;

continue;

}

nextTerm = firstTerm + secondTerm;

firstTerm = secondTerm;

secondTerm = nextTerm;

cout << nextTerm << ", ";

i++;

}

return 0;

}

**Solution 31:**

#include <iostream>

using namespace std;

int findGCD(int a, int b) {

while (b != 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

int main() {

int num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

int gcd = findGCD(num1, num2);

cout << "GCD of " << num1 << " and " << num2 << " is: " << gcd << endl;

return 0;

}

**Solution 32:**

#include <iostream>

using namespace std;

void swapNumbers(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

int main() {

int num1, num2;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

cout << "Before swapping: ";

cout << "num1 = " << num1 << ", num2 = " << num2 << endl;

swapNumbers(num1, num2);

cout << "After swapping: ";

cout << "num1 = " << num1 << ", num2 = " << num2 << endl;

return 0;

}

**Solution 33:**

#include <iostream>

using namespace std;

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

while (number != 0) {

if (number % 2 == 0) {

cout << "The number " << number << " is even." << endl;

} else {

cout << "The number " << number << " is odd." << endl;

}

cout << "Enter another number (or enter 0 to quit): ";

cin >> number;

}

return 0;

}

**Solution 34:**

#include <iostream>

using namespace std;

bool isPrime(int num) {

if (num <= 1) {

return false;

}

for (int i = 2; i\*i <= num; i++) {

if (num % i == 0) {

return false;

}

}

return true;

}

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

if (isPrime(number)) {

cout << number << " is a prime number." << endl;

} else {

cout << number << " is not a prime number." << endl;

}

return 0;

}

**Solution 35:**

#include <iostream>

using namespace std;

int main() {

int n;

double sum = 0.0;

cout << "Enter the value of n: ";

cin >> n;

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

sum += i;

}

double average = sum / n;

cout << "Average of the first " << n << " numbers: " << average << endl;

return 0;

}

**Solution 36:**

#include <iostream>

using namespace std;

int main() {

int num;

unsigned long long factorial = 1;

// Input from user

cout << "Enter a positive integer: ";

cin >> num;

if (num < 0) {

cout << "Error: Factorial of a negative number is undefined." << endl;

}

else {

int i = 1;

do {

factorial \*= i;

i++;

} while (i <= num);

cout << "Factorial of " << num << " = " << factorial << endl;

}

return 0;

}

**Solution 37:**

#include <iostream>

using namespace std;

int main() {

int n, firstTerm = 0, secondTerm = 1, nextTerm;

cout << "Enter the number of terms: ";

cin >> n;

cout << "Fibonacci Series: " << firstTerm << " " << secondTerm << " ";

int i = 3;

do {

nextTerm = firstTerm + secondTerm;

cout << nextTerm << " ";

firstTerm = secondTerm;

secondTerm = nextTerm;

i++;

} while (i <= n);

return 0;

}