**Assignment Number:3**

**Solution 38:**

#include <iostream>

using namespace std;

bool isPrime(int number) {

if (number <= 1) {

return false;

}

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

if (number % i == 0) {

return false;

}

}

return true;

}

int main() {

int num;

cout << "Enter a positive integer: ";

cin >> num;

if (isPrime(num)) {

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

} else {

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

}

return 0;

}

**Solution 39:**

#include <iostream>

using namespace std;

int factorial(int n) {

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

return 1;

}

return n \* factorial(n - 1);

}

int main() {

int number;

cout << "Enter a non-negative integer: ";

cin >> number;

if (number < 0) {

cout << "Error: Factorial is undefined for negative numbers." << endl;

} else {

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

}

return 0;

}

**Solution 40:**

#include <iostream>

using namespace std;

void printFibonacci(int n) {

int first = 0, second = 1, next;

cout << "Fibonacci Series up to " << n << " terms: ";

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

cout << first << " ";

next = first + second;

first = second;

second = next;

}

cout << endl;

}

int main() {

int terms;

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

cin >> terms;

printFibonacci(terms);

return 0;

}

**Solution 41:**

#include <iostream>

using namespace std;

int sumOfDigits(int number) {

int sum = 0;

while (number != 0) {

sum += number % 10;

number /= 10;

}

return sum;

}

int main() {

int num;

cout << "Enter a number: ";

cin >> num;

cout << "Sum of digits of " << num << " is: " << sumOfDigits(num) << endl;

return 0;

}

**Solution 42:**

#include <iostream>

using namespace std;

int reverseNumber(int number) {

int reversedNumber = 0;

while (number != 0) {

int digit = number % 10; // Extract the last digit

reversedNumber = reversedNumber \* 10 + digit;

number /= 10;

}

return reversedNumber;

}

int main() {

int num;

cout << "Enter a number: ";

cin >> num;

cout << "Reverse of " << num << " is: " << reverseNumber(num) << endl;

return 0;

}

**Solution 43:**

#include <iostream>

#include <string>

using namespace std;

string reverseString(string str) {

int length = str.length();

for (int i = 0; i < length / 2; ++i) {

char temp = str[i];

str[i] = str[length - i - 1];

str[length - i - 1] = temp;

}

return str;

}

int main() {

string inputString;

cout << "Enter a string: ";

getline(cin, inputString);

string reversedString = reverseString(inputString);

cout << "Reversed string: " << reversedString << endl;

return 0;

}

**Solution 44:**

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

bool isPalindrome(string str) {

string reversedStr = str;

reverse(reversedStr.begin(), reversedStr.end());

return str == reversedStr;

}

int main() {

string inputString;

// Input a string from the user

cout << "Enter a string: ";

getline(cin, inputString);

if (isPalindrome(inputString)) {

cout << "The string is a palindrome." << endl;

} else {

cout << "The string is not a palindrome." << endl;

}

return 0;

}

**Solution 45:**

#include <iostream>

using namespace std;

double power(double x, int n) {

double result = 1.0;

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

result \*= x;

}

return result;

}

int main() {

double base;

int exponent;

cout << "Enter the base (x): ";

cin >> base;

cout << "Enter the exponent (n): ";

cin >> exponent;

cout << base << " raised to the power of " << exponent << " is: " << power(base, exponent) << endl;

return 0;

}

**Solution 46:**

#include <iostream>

using namespace std;

const int ROWS = 3;

const int COLS = 3;

int sumOfElements(int matrix[ROWS][COLS]) {

int sum = 0;

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

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

sum += matrix[i][j];

}

}

return sum;

}

long long productOfElements(int matrix[ROWS][COLS]) {

long long product = 1;

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

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

product \*= matrix[i][j];

}

}

return product;

}

void transposeMatrix(int matrix[ROWS][COLS]) {

int transposedMatrix[COLS][ROWS];

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

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

transposedMatrix[j][i] = matrix[i][j];

}

}

cout << "Transposed Matrix:" << endl;

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

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

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

}

cout << endl;

}

}

int main() {

int matrix[ROWS][COLS] = {{1, 2, 4},

{3, 1, 6},

{7, 8, 9}};

cout << "Sum of all elements: " << sumOfElements(matrix) << endl;

cout << "Product of all elements: " << productOfElements(matrix) << endl;

transposeMatrix(matrix);

return 0;

}

**Solution 47:**

#include <iostream>

#include <cmath>

using namespace std;

double calculateSeries(int x, int n) {

double sum = 0.0;

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

sum += 1.0 / pow(x, i);

}

return sum;

}

int main() {

int x, n;

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

cin >> x;

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

cin >> n;

double result = calculateSeries(x, n);

cout << "Result of the series: " << result << endl;

return 0;

}

**Solution 50:**

#include <iostream>

using namespace std;

const int MAX\_ROWS = 100;

const int MAX\_COLS = 100;

void multiplyMatrices(int mat1[MAX\_ROWS][MAX\_COLS], int mat2[MAX\_ROWS][MAX\_COLS], int result[MAX\_ROWS][MAX\_COLS], int rows1, int cols1, int rows2, int cols2) {

if (cols1 != rows2) {

cout << "Matrices cannot be multiplied!" << endl;

return;

}

for (int i = 0; i < rows1; ++i) {

for (int j = 0; j < cols2; ++j) {

result[i][j] = 0;

}

}

for (int i = 0; i < rows1; ++i) {

for (int j = 0; j < cols2; ++j) {

for (int k = 0; k < cols1; ++k) {

result[i][j] += mat1[i][k] \* mat2[k][j];

}

}

}

}

void displayMatrix(int mat[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

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

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

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

}

cout << endl;

}

}

int main() {

int mat1[MAX\_ROWS][MAX\_COLS], mat2[MAX\_ROWS][MAX\_COLS], result[MAX\_ROWS][MAX\_COLS];

int rows1, cols1, rows2, cols2;

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

cin >> rows1 >> cols1;

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

for (int i = 0; i < rows1; ++i) {

for (int j = 0; j < cols1; ++j) {

cin >> mat1[i][j];

}

}

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

cin >> rows2 >> cols2;

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

for (int i = 0; i < rows2; ++i) {

for (int j = 0; j < cols2; ++j) {

cin >> mat2[i][j];

}

}

multiplyMatrices(mat1, mat2, result, rows1, cols1, rows2, cols2);

cout << "Product of the matrices:" << endl;

displayMatrix(result, rows1, cols2);

return 0;

}

**Solution 51:**

#include <iostream>

using namespace std;

const int MAX\_ROWS = 100;

const int MAX\_COLS = 100;

void addMatrices(int mat1[MAX\_ROWS][MAX\_COLS], int mat2[MAX\_ROWS][MAX\_COLS], int result[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

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

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

result[i][j] = mat1[i][j] + mat2[i][j];

}

}

}

void displayMatrix(int mat[MAX\_ROWS][MAX\_COLS], int rows, int cols) {

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

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

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

}

cout << endl;

}

}

int main() {

int mat1[MAX\_ROWS][MAX\_COLS], mat2[MAX\_ROWS][MAX\_COLS], result[MAX\_ROWS][MAX\_COLS];

int rows, cols;

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

cin >> rows >> cols;

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

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

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

cin >> mat1[i][j];

}

}

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

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

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

cin >> mat2[i][j];

}

}

addMatrices(mat1, mat2, result, rows, cols);

cout << "Sum of the matrices:" << endl;

displayMatrix(result, rows, cols);

return 0;

}

**Solution 52:**

#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 to find their GCD: ";

cin >> num1 >> num2;

cout << "GCD of " << num1 << " and " << num2 << " is: " << findGCD(num1, num2) << endl;

return 0;

}

**Solution 53:**

#include <iostream>

using namespace std;

double add(double num1, double num2) {

return num1 + num2;

}

double subtract(double num1, double num2) {

return num1 - num2;

}

double multiply(double num1, double num2) {

return num1 \* num2;

}

double divide(double num1, double num2) {

if (num2 == 0) {

cout << "Error: Division by zero!" << endl;

return 0;

}

return num1 / num2;

}

int main() {

char op;

double num1, num2;

cout << "Enter operator (+, -, \*, /): ";

cin >> op;

cout << "Enter two numbers: ";

cin >> num1 >> num2;

switch(op) {

case '+':

cout << "Result: " << add(num1, num2) << endl;

break;

case '-':

cout << "Result: " << subtract(num1, num2) << endl;

break;

case '\*':

cout << "Result: " << multiply(num1, num2) << endl;

break;

case '/':

cout << "Result: " << divide(num1, num2) << endl;

break;

default:

cout << "Invalid operator!" << endl;

}

return 0;

}

**Solution 54:**

#include <iostream>

using namespace std;

int add(int num1, int num2) {

return num1 + num2;

}

int main() {

int num1, num2;

cout << "Enter two numbers to add: ";

cin >> num1 >> num2;

cout << "Sum: " << add(num1, num2) << endl;

return 0;

}