**Assignment Number:5**

**Solution 1:**

#include <iostream>

using namespace std;

void swapArray(int\* arr1, int\* arr2, int size) {

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

int temp = arr1[i];

arr1[i] = arr2[i];

arr2[i] = temp;

}

}

int main() {

int arr1[] = {1, 2, 3, 4, 5};

int arr2[] = {6, 7, 8, 9, 10};

int size = sizeof(arr1) / sizeof(arr1[0]);

cout << "Before swapping:" << endl;

cout << "Array 1: ";

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

cout << arr1[i] << " ";

}

cout << endl << "Array 2: ";

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

cout << arr2[i] << " ";

}

cout << endl;

swapArray(arr1, arr2, size);

cout << "After swapping:" << endl;

cout << "Array 1: ";

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

cout << arr1[i] << " ";

}

cout << endl << "Array 2: ";

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

cout << arr2[i] << " ";

}

cout << endl;

return 0;

}

**Solution 2:**

#include <iostream>

using namespace std;

int countEven(int\* arr, int size) {

int count = 0;

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

if (\*(arr + i) % 2 == 0) {

count++;

}

}

return count;

}

int main() {

int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int size = sizeof(arr) / sizeof(arr[0]);

cout << "Array: ";

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

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

}

cout << endl;

int evenCount = countEven(arr, size);

cout << "Number of even elements: " << evenCount << endl;

return 0;

}

**Solution 3:**

#include <iostream>

#include <cmath>

using namespace std;

double area(double side) {

return side \* side;

}

double area(double length, double width) {

return length \* width;

}

double area(double radius) {

return M\_PI \* radius \* radius;

}

int main() {

double side = 5.0;

double length = 6.0;

double width = 4.0;

double radius = 3.0;

cout << "Area of the square: " << area(side) << endl;

cout << "Area of the rectangle: " << area(length, width) << endl;

cout << "Area of the circle: " << area(radius) << endl;

return 0;

}

**Solution 4:**

#include <iostream>

using namespace std;

void modifyArray(int\* arr, int size) {

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

\*(arr + i) = (\*(arr + i)) \* (\*(arr + i));

}

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

int size = sizeof(arr) / sizeof(arr[0]);

cout << "Array before modification: ";

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

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

}

cout << endl;

modifyArray(arr, size);

cout << "Array after modification: ";

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

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

}

cout << endl;

return 0;

}

**Solution 5:**

#include <iostream>

using namespace std;

void incrementByValue(int& num) {

num++;

}

int main() {

int value = 5;

cout << "Value before increment: " << value << endl;

incrementByValue(value);

cout << "Value after increment: " << value << endl;

return 0;

}

**Solution 6:**

#include <iostream>

using namespace std;

void calculateSumAndAverage(int arr[], int size, int& sum, double& average) {

sum = 0;

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

sum += arr[i];

}

average = static\_cast<double>(sum) / size;

}

int main() {

int arr[] = {1, 2, 3, 4, 5};

int size = sizeof(arr) / sizeof(arr[0]);

int sum;

double average;

calculateSumAndAverage(arr, size, sum, average);

cout << "Sum of elements: " << sum << endl;

cout << "Average of elements: " << average << endl;

return 0;

}

**Solution 7:**

#include <iostream>

#include <vector>

using namespace std;

void inputMatrix(vector<vector<int>>& matrix) {

int rows, cols;

cout << "Enter number of rows: ";

cin >> rows;

cout << "Enter number of columns: ";

cin >> cols;

matrix.resize(rows, vector<int>(cols));

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

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

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

cin >> matrix[i][j];

}

}

}

vector<vector<int>> addMatrices(const vector<vector<int>>& matrix1, const vector<vector<int>>& matrix2) {

int rows = matrix1.size();

int cols = matrix1[0].size();

vector<vector<int>> result(rows, vector<int>(cols));

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

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

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

}

}

return result;

}

void displayMatrix(const vector<vector<int>>& matrix) {

int rows = matrix.size();

int cols = matrix[0].size();

cout << "Resultant Matrix:" << endl;

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

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

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

}

cout << endl;

}

}

int main() {

vector<vector<int>> matrix1, matrix2;

cout << "Enter elements for first matrix:" << endl;

inputMatrix(matrix1);

cout << "Enter elements for second matrix:" << endl;

inputMatrix(matrix2);

if (matrix1.size() != matrix2.size() || matrix1[0].size() != matrix2[0].size()) {

cout << "Matrices must have the same dimensions for addition." << endl;

return 1;

}

vector<vector<int>> result = addMatrices(matrix1, matrix2);

displayMatrix(result);

return 0;

}

**Solution 8:**

#include <iostream>

using namespace std;

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

int temp = a;

a = b;

b = temp;

}

void sortArray(int arr[], int size) {

for (int i = 0; i < size - 1; i++) {

for (int j = 0; j < size - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

int arr[] = {5, 2, 8, 1, 3};

int size = sizeof(arr) / sizeof(arr[0]);

cout << "Array before sorting: ";

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

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

}

cout << endl;

sortArray(arr, size);

cout << "Array after sorting: ";

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

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

}

cout << endl;

return 0;

}

**Solution 9:**

#include <iostream>

using namespace std;

void reverseString(char\* str) {

// Find the length of the string

int length = 0;

while (str[length] != '\0') {

length++;

}

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

char temp = str[i];

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

str[length - i - 1] = temp;

}

}

int main() {

char str[] = "hello";

cout << "String before reversing: " << str << endl;

reverseString(str);

cout << "String after reversing: " << str << endl;

return 0;

}

**Solution 10:**

#include <iostream>

using namespace std;

int factorial(int n) {

if (n == 0)

return 1;

return n \* factorial(n - 1);

}

int main() {

int num;

cout << "Enter a number: ";

cin >> num;

int result = factorial(num);

cout << "Factorial of " << num << " is " << result << endl;

return 0;

}

**Solution 11:**

#include <iostream>

using namespace std;

void removeDuplicates(int arr[], int& size) {

if (size <= 1) // No duplicates if size is 0 or 1

return;

int index = 0;

for (int i = 0; i < size - 1; i++) {

if (arr[i] != arr[i + 1]) {

arr[index++] = arr[i];

}

}

arr[index++] = arr[size - 1];

size = index;

}

int main() {

int arr[] = {1, 2, 2, 3, 4, 4, 5};

int size = sizeof(arr) / sizeof(arr[0]);

cout << "Array before removing duplicates: ";

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

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

}

cout << endl;

removeDuplicates(arr, size);

cout << "Array after removing duplicates: ";

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

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

}

cout << endl;

return 0;

}

**Solution 12:**

#include <iostream>

using namespace std;

if (exponent == 0)

return 1;

if (exponent > 0)

return base \* power(base, exponent - 1);

else

return 1.0 / (base \* power(base, -exponent - 1));

}

int main() {

int base, exponent;

cout << "Enter base: ";

cin >> base;

cout << "Enter exponent: ";

cin >> exponent;

int result = power(base, exponent);

cout << "Result: " << result << endl;

return 0;

}

**Solution 13:**

#include <iostream>

using namespace std;

double computeAverage(int\* arr, int size) {

int sum = 0;

int count = 0;

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

if (arr[i] > 0) {

sum += arr[i];

count++;

}

}

if (count > 0) {

return static\_cast<double>(sum) / count;

} else {

return 0;

}

}

int main() {

int arr[] = {3, -2, 5, -7, 8, 1};

int size = sizeof(arr) / sizeof(arr[0]);

double average = computeAverage(arr, size);

cout << "Average of positive numbers: " << average << endl;

return 0;

}

**Solution 14:**

#include <iostream>

using namespace std;

int sumOfDigits(int num) {

if (num == 0)

return 0;

return num % 10 + sumOfDigits(num / 10);

}

int main() {

int number;

cout << "Enter a number: ";

cin >> number;

int sum = sumOfDigits(number);

cout << "Sum of digits of " << number << " is " << sum << endl;

return 0;

}

**Solution 15:**

#include <iostream>

#include <cstring>

using namespace std;

void concatenateStrings(const char\* str1, const char\* str2, char\* result) {

strcpy(result, str1);

strcat(result, str2);

}

int main() {

const char\* str1 = "Hello, ";

const char\* str2 = "world!";

char result[100];

concatenateStrings(str1, str2, result);

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

return 0;

}

**Solution 16:**

#include <iostream>

using namespace std;

void findMinMax(int\* arr, int size, int\* min, int\* max) {

\*min = \*max = arr[0];

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

if (arr[i] < \*min)

\*min = arr[i];

if (arr[i] > \*max)

\*max = arr[i];

}

}

int main() {

int arr[] = {3, 1, 7, 5, 2, 9};

int size = sizeof(arr) / sizeof(arr[0]);

int min, max;

findMinMax(arr, size, &min, &max);

cout << "Smallest element: " << min << endl;

cout << "Largest element: " << max << endl;

return 0;

}

**Solution 17:**

#include <iostream>

using namespace std;

bool searchElement(const int\* arr, int size, int element, int& index) {

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

if (arr[i] == element) {

index = i;

return true;

}

}

return false;

}

int main() {

int arr[] = {3, 6, 2, 8, 4};

int size = sizeof(arr) / sizeof(arr[0]);

int element, index;

cout << "Enter the element to search: ";

cin >> element;

if (searchElement(arr, size, element, index)) {

cout << "Element " << element << " found at index " << index << endl;

} else {

cout << "Element not found in the array." << endl;

}

return 0;

}