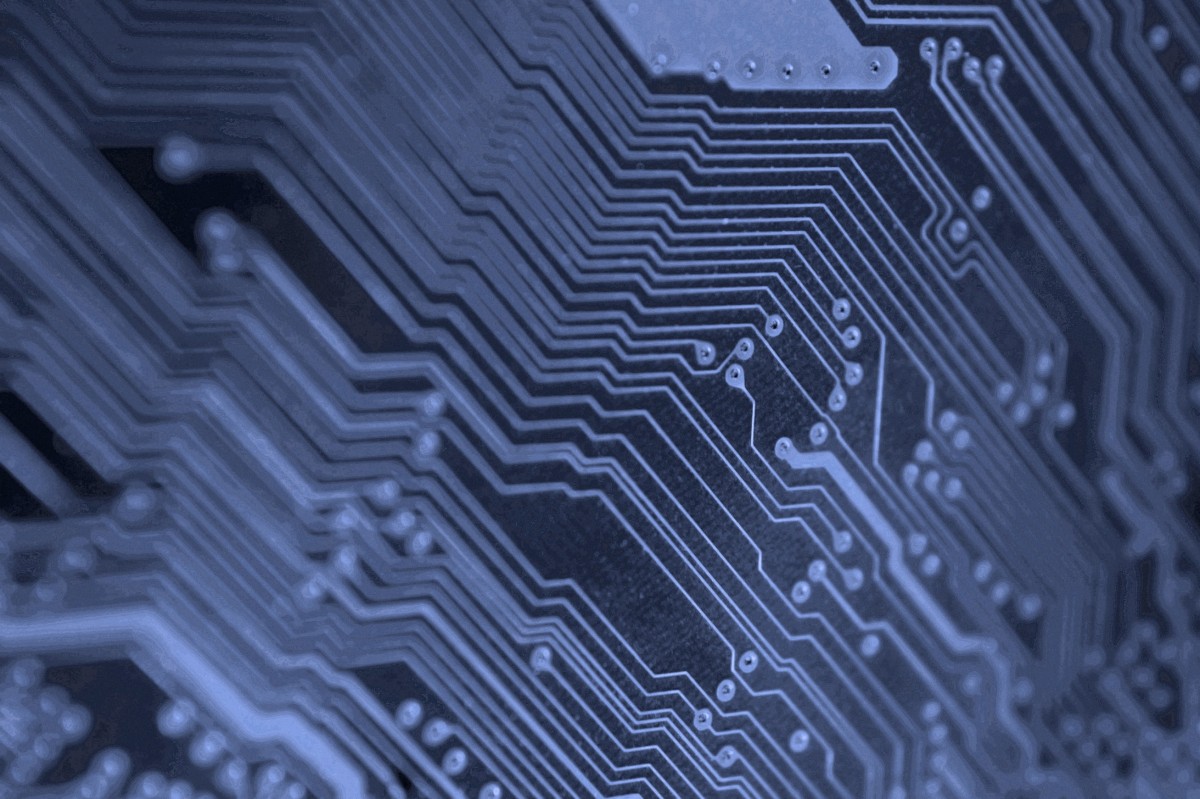
****

**Data Structures**

**Lab 1**

**Name:** Shawaiz Shahid **Roll No.:** 23P-0599

**Date:** 20th Aug2024 **Section:** 3A

**Problem 01**

#include <iostream>

using namespace std;

void swap(int\* a, int\* b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

int main() {

int x, y;

int\* ptrX = &x;

int\* ptrY = &y;

cout << "Enter the first integer: ";

cin >> \*ptrX;

cout << "Enter the second integer: ";

cin >> \*ptrY;

swap(ptrX, ptrY);

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

cout << "First integer: " << \*ptrX << endl;

cout << "Second integer: " << \*ptrY << endl;

system("pause");

return 0;

}

A screenshot of a computer program

Description automatically generated

**Problem 02**

#include <iostream>

using namespace std;

int main() {

int size, valueToFind;

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

cin >> size;

int\* arr = new int[size];

cout << "Enter " << size << " integer values for the array:" << endl;

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

cin >> arr[i];

}

cout << "Enter the value to find: ";

cin >> valueToFind;

bool found = false;

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

if (arr[i] == valueToFind) {

found = true;

break;

}

}

if (found) {

cout << "Value exists in the array." << endl;

}

else {

cout << "Value does not exist in the array." << endl;

}

delete[] arr;

system("pause");

return 0;

}

A black screen with white text

Description automatically generated

**Problem 03**

#include <iostream>

using namespace std;

void Search(int\* arr, int size, int valueToFind) {

bool found = false;

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

if (\*(arr + i) == valueToFind) {

found = true;

break;

}

}

if (found) {

cout << "Value exists in the array." << endl;

}

else {

cout << "Value does not exist in the array." << endl;

}

}

int main() {

int size, valueToFind;

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

cin >> size;

int\* arr = new int[size];

cout << "Enter " << size << " integer values for the array:" << endl;

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

cin >> \*(arr + i);

}

cout << "Enter the value to find: ";

cin >> valueToFind;

Search(arr, size, valueToFind);

delete[] arr;

system("pause");

return 0;

}

A black screen with white text

Description automatically generated

**Problem 04**

#include <iostream>

using namespace std;

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

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

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

cin >> arr[i];

}

}

void removeDuplicates(int\* arr, int& size) {

int newSize = 0;

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

bool duplicate = false;

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

if (arr[i] == arr[j]) {

duplicate = true;

break;

}

}

if (!duplicate) {

arr[newSize++] = arr[i];

}

}

size = newSize;

}

void findCommonElements(int\* arr1, int size1, int\* arr2, int size2, int\* arr3, int& size3) {

size3 = 0;

for (int i = 0; i < size1; i++) {

for (int j = 0; j < size2; j++) {

if (arr1[i] == arr2[j]) {

bool isDuplicate = false;

for (int k = 0; k < size3; k++) {

if (arr3[k] == arr1[i]) {

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

arr3[size3++] = arr1[i];

}

}

}

}

}

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

cout << "Array: ";

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

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

}

cout << endl;

}

int main() {

int size1, size2, size3;

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

cin >> size1;

int\* array1 = new int[size1];

inputValues(array1, size1);

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

cin >> size2;

int\* array2 = new int[size2];

inputValues(array2, size2);

removeDuplicates(array1, size1);

removeDuplicates(array2, size2);

int\* array3 = new int[size1 + size2]; // Max size of common elements

findCommonElements(array1, size1, array2, size2, array3, size3);

cout << "Common elements: ";

printArray(array3, size3);

delete[] array1;

delete[] array2;

delete[] array3;

system("pause");

return 0;

}

A screenshot of a computer screen

Description automatically generated

**Problem 05**

#include <iostream>

using namespace std;

void inputMatrix(int\* matrix, int rows, int cols) {

cout << "Enter the elements of the matrix (" << rows << "x" << cols << "):\n";

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

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

cin >> \*(matrix + i \* cols + j);

}

}

}

void printMatrix(int\* matrix, int rows, int cols) {

cout << "Matrix (" << rows << "x" << cols << "):\n";

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

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

cout << \*(matrix + i \* cols + j) << " ";

}

cout << endl;

}

}

void addMatrices(int\* matrix1, int\* matrix2, int\* resultMatrix, int rows, int cols) {

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

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

\*(resultMatrix + i \* cols + j) = \*(matrix1 + i \* cols + j) + \*(matrix2 + i \* cols + j);

}

}

}

int main() {

int rows = 3, cols = 3;

// memory for three 3x3 matrices

int\* matrix1 = new int[rows \* cols];

int\* matrix2 = new int[rows \* cols];

int\* resultMatrix = new int[rows \* cols];

// values for the two matrices

cout << "Matrix 1:\n";

inputMatrix(matrix1, rows, cols);

cout << "\nMatrix 2:\n";

inputMatrix(matrix2, rows, cols);

addMatrices(matrix1, matrix2, resultMatrix, rows, cols);

// Print matrices

cout << "\nMatrix 1:\n";

printMatrix(matrix1, rows, cols);

cout << "\nMatrix 2:\n";

printMatrix(matrix2, rows, cols);

cout << "\nResult Matrix (After Addition):\n";

printMatrix(resultMatrix, rows, cols);

// Free memory

delete[] matrix1;

delete[] matrix2;

delete[] resultMatrix;

system("pause");

return 0;

}

A screenshot of a computer

Description automatically generated

**Problem 06**

#include <iostream>

using namespace std;

void inputMatrix(int\* matrix, int rows, int cols) {

cout << "Enter the elements of the matrix (" << rows << "x" << cols << "):\n";

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

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

cin >> \*(matrix + i \* cols + j);

}

}

}

void printMatrix(int\* matrix, int rows, int cols) {

cout << "Matrix (" << rows << "x" << cols << "):\n";

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

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

cout << \*(matrix + i \* cols + j) << " ";

}

cout << endl;

}

}

void addMatrices(int\* matrix1, int\* matrix2, int\* resultMatrix, int rows, int cols) {

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

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

\*(resultMatrix + i \* cols + j) = \*(matrix1 + i \* cols + j) + \*(matrix2 + i \* cols + j);

}

}

}

void sortMatrix(int\* matrix, int rows, int cols) {

int size = rows \* cols;

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

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

int\* current = matrix + j;

int\* next = matrix + j + 1;

if (\*current > \*next) {

// Swap using pointer

int temp = \*current;

\*current = \*next;

\*next = temp;

}

}

}

}

int main() {

int rows = 3, cols = 3;

// memory for three 3x3 matrices

int\* matrix1 = new int[rows \* cols];

int\* matrix2 = new int[rows \* cols];

int\* resultMatrix = new int[rows \* cols];

// values for the two matrices

cout << "Matrix 1:\n";

inputMatrix(matrix1, rows, cols);

cout << "\nMatrix 2:\n";

inputMatrix(matrix2, rows, cols);

addMatrices(matrix1, matrix2, resultMatrix, rows, cols);

// Sort the result matrix

sortMatrix(resultMatrix, rows, cols);

// Print matrices

cout << "\nMatrix 1:\n";

printMatrix(matrix1, rows, cols);

cout << "\nMatrix 2:\n";

printMatrix(matrix2, rows, cols);

cout << "\nResult Matrix (After Addition and Sorting):\n";

printMatrix(resultMatrix, rows, cols);

// Free memory

delete[] matrix1;

delete[] matrix2;

delete[] resultMatrix;

system("pause");

return 0;

}

A screenshot of a computer program

Description automatically generated