





Q5 #include <iostream>

using namespace std;

void bubbleSort(int arr[], int n) {

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

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

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

// Swap arr[j] and arr[j+1]

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

int n;

// Input number of elements

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

cin >> n;

int arr[n];

// Input array elements

cout << "Enter the elements: ";

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

cin >> arr[i];

}

// Sort the array

bubbleSort(arr, n);

// Display sorted array

cout << "Sorted Array: ";

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

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

}

return 0;

}

OUTPUT:

Enter the number of elements: 5

Enter the elements: 12 4 7 9 1

Sorted Array: 1 4 7 9 12

Q6:

#include <iostream>

using namespace std;

int linearSearch(int arr[], int n, int key) {

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

if (arr[i] == key) {

return i; }

}

return -1;

}

int main() {

int n, key;

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

cin >> n;

int arr[n];

cout << "Enter the elements: ";

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

cin >> arr[i];

}

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

cin >> key;

int result = linearSearch(arr, n, key);

if (result != -1) {

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

} else {

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

}

return 0;

}

OUTPUT:

Enter the number of elements: 5

Enter the elements: 10 20 30 40 50

Enter the element to search: 30

Element found at index 2

Q7:

#include <iostream>

using namespace std;

int removeDuplicates(int arr[], int n) {

int index = 0;

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

bool isDuplicate = false;

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

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

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

arr[index] = arr[i];

index++;

}

}

return index;

}

int main() {

int n;

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

cin >> n;

int arr[n];

cout << "Enter the elements: ";

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

cin >> arr[i];

}

int newSize = removeDuplicates(arr, n);

cout << "Array after removing duplicates: ";

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

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

}

return 0;

}

OUTPUT:

**Enter the number of elements: 7**

**Enter the elements: 1 2 3 2 4 1 5**

**Array after removing duplicates: 1 2 3 4 5**

**Q8:**

#include <iostream>

#include <unordered\_set>

using namespace std;

int longestConsecutiveSubsequence(int arr[], int n) {

unordered\_set<int> numSet(arr, arr + n);

int longestStreak = 0;

for (int num : numSet) {

if (numSet.find(num - 1) == numSet.end()) {

int currentNum = num;

int currentStreak = 1;

while (numSet.find(currentNum + 1) != numSet.end()) {

currentNum++;

currentStreak++;

}

longestStreak = max(longestStreak, currentStreak);

}

}

return longestStreak;

}

int main() {

int n;

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

cin >> n;

int arr[n];

cout << "Enter the elements: ";

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

cin >> arr[i];

}

int result = longestConsecutiveSubsequence(arr, n);

cout << "Length of the longest consecutive subsequence: " << result << endl;

return 0;

}

OUTPUT:

Enter the number of elements: 6

Enter the elements: 1 9 3 10 2 20

Length of the longest consecutive subsequence: 3

Q9:

#include <iostream>

using namespace std;

int main() {

int rows, cols;

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

cin >> rows >> cols;

int matrix[rows][cols], transpose[cols][rows];

cout << "Enter the elements of the matrix:\n";

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

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

cin >> matrix[i][j];

}

}

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

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

transpose[j][i] = matrix[i][j]; // Swap rows and columns

}

}

cout << "Transpose of the matrix:\n";

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

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

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

}

cout << endl;

}

return 0;

}

OUTPUT:

Enter the number of rows and columns: 2 3

Enter the elements of the matrix:

1 2 3

4 5 6

Transpose of the matrix:

1 4

2 5

3 6

Q10:

#include <iostream>

using namespace std;

int main() {

int r1, c1, r2, c2;

cout << "Enter rows and columns for first matrix: ";

cin >> r1 >> c1;

cout << "Enter rows and columns for second matrix: ";

cin >> r2 >> c2;

if (c1 != r2) {

cout << "Matrix multiplication not possible! Columns of first matrix must equal rows of second matrix." << endl;

return 0;

}

int mat1[r1][c1], mat2[r2][c2], result[r1][c2];

cout << "Enter elements of first matrix:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c1; j++) {

cin >> mat1[i][j];

}

}

cout << "Enter elements of second matrix:\n";

for (int i = 0; i < r2; i++) {

for (int j = 0; j < c2; j++) {

cin >> mat2[i][j];

}

}

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

result[i][j] = 0;

}

}

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

for (int k = 0; k < c1; k++) {

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

}

}

}

cout << "Resultant Matrix after multiplication:\n";

for (int i = 0; i < r1; i++) {

for (int j = 0; j < c2; j++) {

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

}

cout << endl;

}

return 0;

}

OUTPUT:

Enter rows and columns for first matrix: 2 3

Enter rows and columns for second matrix: 3 2

Enter elements of first matrix:

1 2 3

4 5 6

Enter elements of second matrix:

7 8

9 10

11 12

Resultant Matrix after multiplication:

58 64

139 154

Q11:

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter the size of the square matrix (n x n): ";

cin >> n;

int matrix[n][n];

cout << "Enter the elements of the matrix:\n";

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

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

cin >> matrix[i][j];

}

}

int primarySum = 0, secondarySum = 0;

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

primarySum += matrix[i][i];

secondarySum += matrix[i][n - i - 1];

}

cout << "Sum of Primary Diagonal: " << primarySum << endl;

cout << "Sum of Secondary Diagonal: " << secondarySum << endl;

return 0;

}

OUTPUT:

Enter the size of the square matrix (n x n): 3

Enter the elements of the matrix:

1 2 3

4 5 6

7 8 9

Sum of Primary Diagonal: 15

Sum of Secondary Diagonal: 15

Q12:

#include <iostream>

using namespace std;

int main() {

int rows, cols;

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

cin >> rows >> cols;

int matrix[rows][cols];

cout << "Enter the elements of the matrix:\n";

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

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

cin >> matrix[i][j];

}

}

int maxSum = 0, maxRowIndex = 0;

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

int rowSum = 0;

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

rowSum += matrix[i][j]; }

if (rowSum > maxSum) {

maxSum = rowSum;

maxRowIndex = i;

}

}

cout << "Row with the maximum sum: " << maxRowIndex << " (Sum = " << maxSum << ")" << endl;

return 0;

}

OUTPUT:

Enter the number of rows and columns: 3 3

Enter the elements of the matrix:

1 2 3

4 5 6

7 8 9

Row with the maximum sum: 2 (Sum = 24)

Q13:

#include <iostream>

using namespace std;

bool searchInSortedMatrix(int matrix[][100], int rows, int cols, int target) {

int i = 0, j = cols - 1;

while (i < rows && j >= 0) {

if (matrix[i][j] == target) {

cout << "Element found at position: (" << i << ", " << j << ")\n";

return true;

}

else if (matrix[i][j] > target) {

j--;

}

else {

i++;

}

}

cout << "Element not found in the matrix.\n";

return false;

}

int main() {

int rows, cols, target;

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

cin >> rows >> cols;

int matrix[100][100];

cout << "Enter the sorted matrix elements:\n";

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

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

cin >> matrix[i][j];

}

}

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

cin >> target;

searchInSortedMatrix(matrix, rows, cols, target);

return 0;

}

OUTPUT:

Enter the number of rows and columns: 4 4

Enter the sorted matrix elements:

1 4 7 11

2 5 8 12

3 6 9 16

10 13 14 17

Enter the target element to search: 6

Element found at position: (2, 1)

Q14:

#include <iostream>

using namespace std;

void printBoundaryElements(int matrix[][100], int rows, int cols) {

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

cout << matrix[0][j] << " ";

}

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

cout << matrix[i][cols - 1] << " ";

}

if (rows > 1) {

for (int j = cols - 1; j >= 0; j--) {

cout << matrix[rows - 1][j] << " ";

}

}

if (cols > 1) {

for (int i = rows - 2; i > 0; i--) {

cout << matrix[i][0] << " ";

}

}

cout << endl;

}

int main() {

int rows, cols;

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

cin >> rows >> cols;

int matrix[100][100];

cout << "Enter the matrix elements:\n";

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

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

cin >> matrix[i][j];

}

}

cout << "Boundary elements of the matrix: ";

printBoundaryElements(matrix, rows, cols);

return 0;

}

OUTPUT:

Enter the number of rows and columns: 4 4

Enter the matrix elements:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Boundary elements of the matrix: 1 2 3 4 8 12 16 15 14 13 9 5

Q15:

#include <iostream>

using namespace std;

void findSaddlePoint(int matrix[][100], int rows, int cols) {

bool found = false;

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

int minRow = matrix[i][0], colIndex = 0;

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

if (matrix[i][j] < minRow) {

minRow = matrix[i][j];

colIndex = j;

}

}

bool isSaddlePoint = true;

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

if (matrix[k][colIndex] > minRow) {

isSaddlePoint = false;

break;

}

}

if (isSaddlePoint) {

cout << "Saddle point found at (" << i << ", " << colIndex << ") with value: " << minRow << endl;

found = true;

}

}

if (!found) {

cout << "No saddle point found in the matrix.\n";

}

}

int main() {

int rows, cols;

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

cin >> rows >> cols;

int matrix[100][100];

cout << "Enter the matrix elements:\n";

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

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

cin >> matrix[i][j];

}

}

findSaddlePoint(matrix, rows, cols);

return 0;

}

OUTPUT:

Enter the number of rows and columns: 3 3

Enter the matrix elements:

1 2 3

4 5 6

7 8 9

Saddle point found at (1, 0) with value: 4

Q1:

#include <iostream>

using namespace std;

void printButterfly(int n) {

// Upper Half

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

// Left stars

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

cout << "\*";

}

// Spaces

for (int j = 1; j <= 2 \* (n - i); j++) {

cout << " ";

}

// Right stars

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

cout << "\*";

}

cout << endl;

}

// Lower Half

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

// Left stars

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

cout << "\*";

}

// Spaces

for (int j = 1; j <= 2 \* (n - i); j++) {

cout << " ";

}

// Right stars

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

cout << "\*";

}

cout << endl;

}

}

int main() {

int n;

// Input size of butterfly

cout << "Enter the size of butterfly pattern: ";

cin >> n;

// Print butterfly pattern

printButterfly(n);

return 0;

}

OUTPUT:

Enter the size of butterfly pattern: 4

\* \*

\*\* \*\*

\*\*\* \*\*\*

\*\*\*\*\*\*\*\*

\*\*\* \*\*\*

\*\* \*\*

\* \*