Assignment 2- (divanshi garg)

Q1. #include <iostream>

using namespace std;

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

    int left = 0;

    int right = n - 1;

    while (left <= right) {

        int mid = left + (right - left) / 2;

        if (arr[mid] == key)

            return mid;

        if (arr[mid] < key)

            left = mid + 1;

        else

            right = mid - 1;

    }

    return -1;

}

int main() {

    int n, key;

    cout << "Enter size of array: ";

    cin >> n;

    int arr[n];

    cout << "Enter sorted elements: ";

    for (int i = 0; i < n; i++)

        cin >> arr[i];

    cout << "Enter key to search: ";

    cin >> key;

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

    if (result != -1)

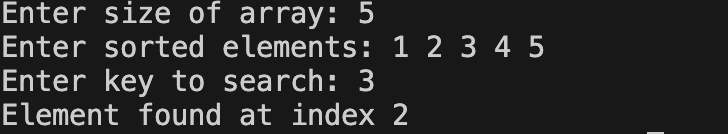
        cout << "Element found at index " << result << endl;

    else

        cout << "Element not found" << endl;

    return 0;

}



Q2)

#include <iostream>

using namespace std;

int main() {

    int arr[100],n;

    cout<<"enter size of array: ";

    cin>>n;

    cout<<"enter elements: ";

    for(int i=0;i<n;i++){

        cin>>arr[i];

    }

    cout<<"the unsorted array is: ";

    for(int i=0;i<n;i++){

        cout<<arr[i];

    }

    for (int i = 0; i < n - 1; i++) {

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

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

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

    cout << "Sorted array: ";

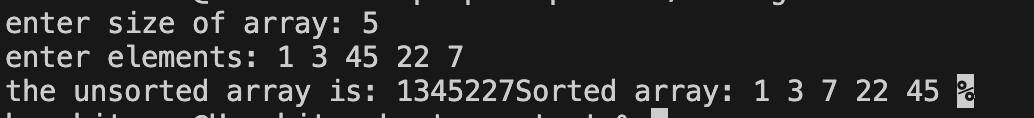
    for (int i = 0; i < n; i++) {

        cout << arr[i] << " ";

    }

    return 0;

}



Q3

#include <iostream>

using namespace std;

int main() {

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

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

    int expectedSum = (n + 1) \* (n + 2) / 2;

    int actualSum = 0;

    for (int i = 0; i < n; i++) {

        actualSum += arr[i];

    }

    int missingNumber = expectedSum - actualSum;

    cout << "Missing number is: " << missingNumber << endl;

    return 0;

}



Q4 a).

#include<iostream>

#include<cstring>

using namespace std;

int main(){

    char s1[50],s2[50];

    cout<<"enter the first string"<<" ";

    cin>>s1;

    cout<<"enter the second string"<<" ";

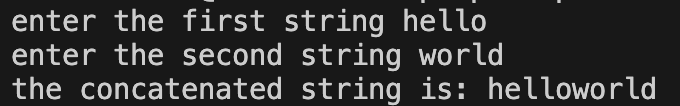
    cin>>s2;

    strcat(s1,s2);

    cout<<"the concatenated string is: "<<s1<<endl;

    return 0;

}



Q4 b)

#include<iostream>

#include<cstring>

using namespace std;

int main(){

     char str[100];

     cout<<"enter the string: ";

     cin>>str;

     int len = strlen(str);

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

            int temp=str[i];

            str[i]=str[j];

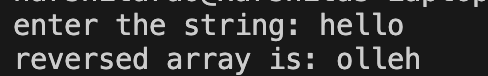
            str[j]=temp;

     }

     cout<<"reversed array is: "<<str<<endl;

     return 0;

    }



Q4 c) #include <iostream>

#include <string>

using namespace std;

int main() {

    string str, result = " ";

    cout << "Enter the string: ";

    cin>>str;

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

if (str[i] != 'a' && str[i] != 'e' && str[i] != 'i' && str[i] != 'o' && str[i] != 'u' &&str[i] != 'A' && str[i] != 'E' && str[i] != 'I' && str[i] != 'O' && str[i] != 'U') {

            result += str[i];

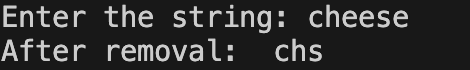
        }

    }

    cout << "After removal: " << result << endl;

    return 0;

}



Q4 d)

#include <iostream>

#include <string>

using namespace std;

int main() {

    int n;

    cout << "Enter number of strings: ";

    cin >> n;

    string arr[100];

    cout << "Enter strings:\n";

    for (int i = 0; i < n; i++) {

        cin >> arr[i];

    }

    for (int i = 0; i < n - 1; i++) {

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

            if (arr[i] > arr[j]) {

                string temp = arr[i];

                arr[i] = arr[j];

                arr[j] = temp;

            }

        }

    }

    cout << "Strings in alphabetical order:\n";

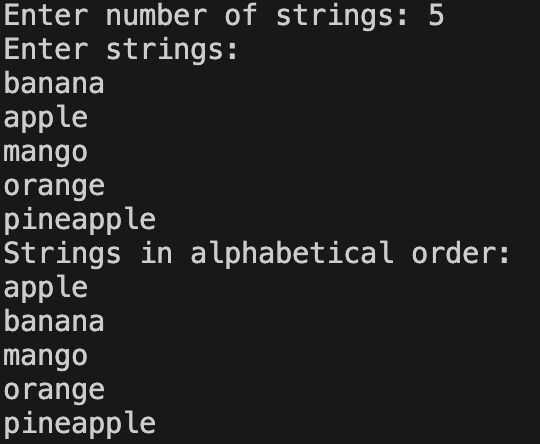
    for (int i = 0; i < n; i++) {

        cout << arr[i] << endl;

    }

    return 0;

}



Q4 e)

#include <iostream>

using namespace std;

int main() {

    char ch;

    cout << "Enter an uppercase character: ";

    cin >> ch;

    if (ch >= 'A' && ch <= 'Z') {

        ch = ch + 32;

        cout << "Lowercase: " << ch << endl;

    } else {

        cout << "The entered character is not uppercase." << endl;

    }

    return 0;

}

Q5.a)

#include <iostream>

using namespace std;

class DiagonalMatrix {

int A[100];

int n;

public:

void setSize(int size) {

n = size;

for (int i = 0; i < n; i++) A[i] = 0;

}

void set(int i, int j, int val) {

if (i == j)

A[i] = val;

}

int get(int i, int j) {

if (i == j)

return A[i];

else

return 0;

}

void display() {

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

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

if (i == j)

cout << A[i] << " ";

else

cout << "0 ";

}

cout << endl;

}

}

};

int main() {

int n;

cout << "Enter dimension of diagonal matrix: ";

cin >> n;

DiagonalMatrix dm;

dm.setSize(n);

cout << "Enter diagonal elements:\n";

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

int val;

cin >> val;

dm.set(i, i, val);

}

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

dm.display();

return 0;

}

Q5.b)

#include <iostream>

using namespace std;

class TriDiagonal {

int A[100];

int n;

public:

void setSize(int size) {

n = size;

for (int i = 0; i < 3\*n-2; i++) A[i] = 0;

}

void set(int i, int j, int val) {

if (i - j == 1)

A[i - 1] = val;

else if (i == j)

A[n - 1 + i - 1] = val;

else if (j - i == 1)

A[2\*n - 1 + i - 1] = val;

}

int get(int i, int j) {

if (i - j == 1)

return A[i - 1];

else if (i == j)

return A[n - 1 + i - 1];

else if (j - i == 1)

return A[2\*n - 1 + i - 1];

else

return 0;

}

void display() {

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

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

cout << get(i, j) << " ";

}

cout << endl;

}

}

};

int main() {

int n;

cout << "Enter dimension of tri-diagonal matrix: ";

cin >> n;

TriDiagonal td;

td.setSize(n);

cout << "Enter elements row-wise:\n";

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

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

int val;

cin >> val;

td.set(i, j, val);

}

}

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

td.display();

return 0;

}

Q5.c)

#include <iostream>

using namespace std;

class LowerTriangular {

int A[100];

int n;

public:

void setSize(int size) {

n = size;

for (int i = 0; i < (n\*(n+1))/2; i++) A[i] = 0;

}

void set(int i, int j, int val) {

if (i >= j) {

int index = (i\*(i-1))/2 + (j-1);

A[index] = val;

}

}

int get(int i, int j) {

if (i >= j) {

int index = (i\*(i-1))/2 + (j-1);

return A[index];

}

else

return 0;

}

void display() {

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

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

cout << get(i, j) << " ";

}

cout << endl;

}

}

};

int main() {

int n;

cout << "Enter dimension of lower triangular matrix: ";

cin >> n;

LowerTriangular lt;

lt.setSize(n);

cout << "Enter elements row-wise:\n";

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

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

int val;

cin >> val;

lt.set(i, j, val);

}

}

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

lt.display();

return 0;

Q5.d)

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter dimension of upper triangular matrix: ";

cin >> n;

int size = (n \* (n + 1)) / 2;

int A[size];

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

A[i] = 0;

}

cout << "Enter elements row-wise:\n";

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

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

int val;

cin >> val;

if (i <= j) {

int index = (i - 1) \* n - ((i - 2) \* (i - 1)) / 2 + (j - i);

A[index] = val;

}

}

}

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

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

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

if (i <= j) {

int index = (i - 1) \* n - ((i - 2) \* (i - 1)) / 2 + (j - i);

cout << A[index] << " ";

} else {

cout << "0 ";

}

}

cout << endl;

}

return 0;

}

Q5.e)

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter dimension of symmetric matrix: ";

cin >> n;

int size = (n \* (n + 1)) / 2;

int A[size];

cout << "Enter elements row-wise:\n";

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

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

int val;

cin >> val;

if (i >= j) {

int index = (i \* (i - 1)) / 2 + (j - 1);

A[index] = val;

}

}

}

cout << "\nSymmetric Matrix is:\n";

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

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

if (i >= j) {

int index = (i \* (i - 1)) / 2 + (j - 1);

cout << A[index] << " ";

} else {

int index = (j \* (j - 1)) / 2 + (i - 1);

cout << A[index] << " ";

}

}

cout << endl;

}

return 0;

}

Q6.a)

#include<iostream>

using namespace std;

int main(){

int rows, cols, n;

cout << "Enter rows and cols: ";

cin >> rows >> cols;

cout << "Enter the number of non-zero elements: ";

cin >> n;

int matrix[n+1][3];

matrix[0][0] = rows;

matrix[0][1] = cols;

matrix[0][2] = n;

cout << "Enter the row, col, value of non-zero elements:\n";

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

cin >> matrix[i][0] >> matrix[i][1] >> matrix[i][2];

}

cout << "Original Triplet Representation:\n";

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

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

<< matrix[i][1] << " "

<< matrix[i][2] << endl;

}

int transpose[n+1][3];

transpose[0][0] = matrix[0][1];

transpose[0][1] = matrix[0][0];

transpose[0][2] = matrix[0][2];

int k = 1;

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

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

if(matrix[j][1] == i){

transpose[k][0] = matrix[j][1];

transpose[k][1] = matrix[j][0];

transpose[k][2] = matrix[j][2];

k++;

}

}

}

cout << "Transpose Triplet Representation:\n";

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

cout << transpose[i][0] << " "

<< transpose[i][1] << " "

<< transpose[i][2] << endl;

}

return 0;

}

Q6.b)

#include<iostream>

using namespace std;

int main() {

int r1, c1, r2, c2, n1, n2;

cout << "Enter rows and cols of A: ";

cin >> r1 >> c1;

cout << "Enter the number of non-zero elements in A: ";

cin >> n1;

cout << "Enter rows and cols of B: ";

cin >> r2 >> c2;

cout << "Enter the number of non-zero elements in B: ";

cin >> n2;

int A[n1+1][3];

A[0][0] = r1; A[0][1] = c1; A[0][2] = n1;

for(int i = 1; i <= n1; i++) {

cin >> A[i][0] >> A[i][1] >> A[i][2];

}

int B[n2+1][3];

B[0][0] = r2; B[0][1] = c2; B[0][2] = n2;

for(int i = 1; i <= n2; i++) {

cin >> B[i][0] >> B[i][1] >> B[i][2];

}

if(r1 != r2 || c1 != c2) {

cout << "Addition not possible";

return 0;

}

int C[n1+n2+1][3];

C[0][0] = r1;

C[0][1] = c1;

int i = 1, j = 1, k = 1;

// Merge like merge sort

while(i <= n1 && j <= n2) {

if(A[i][0] < B[j][0] || (A[i][0] == B[j][0] && A[i][1] < B[j][1])) {

C[k][0] = A[i][0];

C[k][1] = A[i][1];

C[k][2] = A[i][2];

i++; k++;

}

else if(B[j][0] < A[i][0] || (A[i][0] == B[j][0] && B[j][1] < A[i][1])) {

C[k][0] = B[j][0];

C[k][1] = B[j][1];

C[k][2] = B[j][2];

j++; k++;

}

else {

int sum = A[i][2] + B[j][2];

if(sum != 0) {

C[k][0] = A[i][0];

C[k][1] = A[i][1];

C[k][2] = sum;

k++;

}

i++; j++;

}

}

while(i <= n1) {

C[k][0] = A[i][0];

C[k][1] = A[i][1];

C[k][2] = A[i][2];

i++; k++;

}

while(j <= n2) {

C[k][0] = B[j][0];

C[k][1] = B[j][1];

C[k][2] = B[j][2];

j++; k++;

}

C[0][2] = k-1;

cout << "\nResult:\n";

for(int x = 0; x <= C[0][2]; x++) {

cout << C[x][0] << " " << C[x][1] << " " << C[x][2] << endl;

}

return 0;

}

Q6.c)

#include <iostream>

using namespace std;

int main() {

int r1, c1, n1, r2, c2, n2;

cout << "Enter rows and cols of Matrix A: ";

cin >> r1 >> c1;

cout << "Enter number of non-zero elements in A: ";

cin >> n1;

int A[n1+1][3];

A[0][0] = r1; A[0][1] = c1; A[0][2] = n1;

cout << "Enter row col value for A:\n";

for (int i = 1; i <= n1; i++)

cin >> A[i][0] >> A[i][1] >> A[i][2];

cout << "Enter rows and cols of Matrix B: ";

cin >> r2 >> c2;

cout << "Enter number of non-zero elements in B: ";

cin >> n2;

int B[n2+1][3];

B[0][0] = r2; B[0][1] = c2; B[0][2] = n2;

cout << "Enter row col value for B:\n";

for (int i = 1; i <= n2; i++)

cin >> B[i][0] >> B[i][1] >> B[i][2];

if (c1 != r2) {

cout << "Multiplication not possible!";

return 0;

}

int result[r1][c2];

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

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

result[i][j] = 0;

for (int i = 1; i <= n1; i++) {

for (int j = 1; j <= n2; j++) {

if (A[i][1] == B[j][0]) {

result[A[i][0]][B[j][1]] += A[i][2] \* B[j][2];

}

}

}

int C[r1\*c2 + 1][3];

int k = 1;

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

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

if (result[i][j] != 0) {

C[k][0] = i;

C[k][1] = j;

C[k][2] = result[i][j];

k++;

}

}

}

C[0][0] = r1;

C[0][1] = c2;

C[0][2] = k-1;

cout << "Result Matrix :\n";

for (int i = 0; i <= C[0][2]; i++) {

cout << C[i][0] << " " << C[i][1] << " " << C[i][2] << endl;

}

return 0;

}

Q7)

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of array: ";

cin >> n;

int A[n];

cout << "Enter elements: ";

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

cin >> A[i];

}

int count = 0;

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

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

if (A[i] > A[j]) {

count++;

}

}

}

cout << "Number of inversions = " << count << endl;

return 0;

}

Q8)

#include<iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of array: ";

cin >> n;

int arr[n];

cout << "Enter elements: ";

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

cin >> arr[i];

}

int distinctCount = 0;

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

bool isDistinct = true;

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

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

isDistinct = false;

break;

}

}

if (isDistinct) {

distinctCount++;

}

}

cout << "Total number of distinct elements = " << distinctCount << endl;

return 0;

}