ASSIGNMENT 2

ADHYA RAZDAN, 1024170337,2Q32

**Q1. BINARY SEARCH**

#include <iostream>

using namespace std;

int main() {

    int n, key;

    cout << "Enter number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter " << n << " elements in sorted order: ";

    for (int i = 0; i < n; i++) {

        cin >> arr[i];

    }

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

    cin >> key;

    int low = 0, high = n - 1, mid;

    bool found = false;

    while (low <= high) {

        mid = (low + high) / 2;

        if (arr[mid] == key) {

            cout << "Element found at position " << mid + 1 << endl;

            found = true;

            break;

        }

        else if (arr[mid] < key) {

            low = mid + 1; // Search right half

        }

        else {

            high = mid - 1; // Search left half

        }

    }

    if (!found) {

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

    }

    return 0;

}



**Q2. BUBBLE SORT**



**Q3. FIND MISSING NUMBER IN SORTED ARRAY**

**Q4. STRING RELATED PROBLEMS**

#include <iostream>

#include <cstring>

using namespace std;

#define MAX 100

#define STR\_COUNT 5

void concatenateStrings(char str1[], char str2[]) {

    strcat(str1, str2);

    cout << "Concatenated string: " << str1 << endl;

}

void reverseString(char str[]) {

    int len = strlen(str);

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

        char temp = str[i];

        str[i] = str[len - i - 1];

        str[len - i - 1] = temp;

    }

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

}

void deleteVowels(char str[]) {

    char result[MAX];

    int j = 0;

    for (int i = 0; str[i] != '\0'; i++) {

        char ch = tolower(str[i]);

        if (!(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u')) {

            result[j++] = str[i];

        }

    }

    result[j] = '\0';

    cout << "String without vowels: " << result << endl;

}

void sortStrings(char arr[STR\_COUNT][MAX], int n) {

    char temp[MAX];

    for (int i = 0; i < n - 1; i++) {

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

            if (strcmp(arr[i], arr[j]) > 0) {

                strcpy(temp, arr[i]);

                strcpy(arr[i], arr[j]);

                strcpy(arr[j], temp);

            }

        }

    }

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

    for (int i = 0; i < n; i++) {

        cout << arr[i] << endl;

    }

}

void toLowerCase(char str[]) {

    for (int i = 0; str[i] != '\0'; i++) {

        str[i] = tolower(str[i]);

    }

    cout << "Lowercase string: " << str << endl;

}

int main() {

    int choice;

    char str1[MAX], str2[MAX];

    char strings[STR\_COUNT][MAX];

    int n;

    do {

        cout << "\n---- STRING OPERATIONS ----\n";

        cout << "1. Concatenate Strings\n";

        cout << "2. Reverse String\n";

        cout << "3. Delete Vowels\n";

        cout << "4. Sort Strings Alphabetically\n";

        cout << "5. Convert Uppercase to Lowercase\n";

        cout << "6. Exit\n";

        cout << "Enter your choice: ";

        cin >> choice;

        switch (choice) {

            case 1:

                cout << "Enter first string: ";

                cin.getline(str1, MAX);

                cout << "Enter second string: ";

                cin.getline(str2, MAX);

                concatenateStrings(str1, str2);

                break;

            case 2:

                cout << "Enter a string: ";

                cin.getline(str1, MAX);

                reverseString(str1);

                break;

            case 3:

                cout << "Enter a string: ";

                cin.getline(str1, MAX);

                deleteVowels(str1);

                break;

            case 4:

                cout << "Enter number of strings (max " << STR\_COUNT << "): ";

                cin >> n;

                cin.ignore();

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

                    cout << "Enter string " << i + 1 << ": ";

                    cin.getline(strings[i], MAX);

                }

                sortStrings(strings, n);

                break;

            case 5:

                cout << "Enter a string: ";

                cin.getline(str1, MAX);

                toLowerCase(str1);

                break;

            case 6:

                cout << "Exiting...\n";

                break;

            default:

                cout << "Invalid choice!\n";

        }

    } while (choice != 6);

    return 0;

}

**Q5.**

**DIAGONAL MATRIX**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of square matrix: ";

cin >> n;

int mat[n][n];

cout << "Enter elements:\n";

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

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

cin >> mat[i][j];

bool isDiagonal = true;

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

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

if(i != j && mat[i][j] != 0)

isDiagonal = false;

if(isDiagonal)

cout << "Matrix is Diagonal\n";

else

cout << "Matrix is NOT Diagonal\n";

return 0;

}

**TRI-DIAGONAL MATRIX**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of square matrix: ";

cin >> n;

int mat[n][n];

cout << "Enter elements:\n";

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

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

cin >> mat[i][j];

bool isTriDiagonal = true;

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

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

if(abs(i-j) > 1 && mat[i][j] != 0)

isTriDiagonal = false;

if(isTriDiagonal) cout << "Matrix is Tri-diagonal\n";

else cout << "Matrix is NOT Tri-diagonal\n";

return 0;

}

**LOWER TRIANGULAR MATRIX**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of square matrix: ";

cin >> n;

int mat[n][n];

cout << "Enter elements:\n";

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

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

cin >> mat[i][j];

bool isLower = true;

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

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

if(mat[i][j] != 0)

isLower = false;

if(isLower) cout << "Matrix is Lower Triangular\n";

else cout << "Matrix is NOT Lower Triangular\n";

return 0;

}

**UPPER TRIANGULAR MATRIX**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of square matrix: ";

cin >> n;

int mat[n][n];

cout << "Enter elements:\n";

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

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

cin >> mat[i][j];

bool isUpper = true;

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

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

if(mat[i][j] != 0)

isUpper = false;

if(isUpper) cout << "Matrix is Upper Triangular\n";

else cout << "Matrix is NOT Upper Triangular\n";

return 0;

}

**SYMMETRIC MATRIX**

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter size of square matrix: ";

cin >> n;

int mat[n][n];

cout << "Enter elements:\n";

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

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

cin >> mat[i][j];

bool isSymmetric = true;

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

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

if(mat[i][j] != mat[j][i])

isSymmetric = false;

if(isSymmetric) cout << "Matrix is Symmetric\n";

else cout << "Matrix is NOT Symmetric\n";

return 0;

}

**Q6.**

#include <iostream>

using namespace std;

struct Triplet {

int row;

int col;

int value;

};

// Function to print sparse matrix

void printSparse(Triplet arr[], int n) {

cout << "Row\tCol\tValue\n";

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

cout << arr[i].row << "\t" << arr[i].col << "\t" << arr[i].value << endl;

}

}

// Transpose of sparse matrix

void transpose(Triplet arr[], Triplet t[], int n) {

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

t[i].row = arr[i].col;

t[i].col = arr[i].row;

t[i].value = arr[i].value;

}

}

// Addition of two sparse matrices

int addSparse(Triplet a[], int n1, Triplet b[], int n2, Triplet result[]) {

int i = 0, j = 0, k = 0;

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

if(a[i].row < b[j].row || (a[i].row == b[j].row && a[i].col < b[j].col))

result[k++] = a[i++];

else if(b[j].row < a[i].row || (b[j].row == a[i].row && b[j].col < a[i].col))

result[k++] = b[j++];

else {

result[k] = a[i];

result[k].value = a[i].value + b[j].value;

i++; j++; k++;

}

}

while(i < n1) result[k++] = a[i++];

while(j < n2) result[k++] = b[j++];

return k;

}

// Multiplication of two sparse matrices

int multiplySparse(Triplet a[], int n1, int aRows, int aCols,

Triplet b[], int n2, int bRows, int bCols,

Triplet result[]) {

if(aCols != bRows) {

cout << "Multiplication not possible!\n";

return 0;

}

int k = 0;

for(int i = 0; i < n1; i++) {

for(int j = 0; j < n2; j++) {

if(a[i].column == b[j].row) {

// Check if position already exists

int found = -1;

for(int m = 0; m < k; m++)

if(result[m].row == a[i].row && result[m].column == b[j].column)

found = m;

if(found != -1)

result[found].value += a[i].value \* b[j].value;

else {

result[k].row = a[i].row;

result[k].column = b[j].column;

result[k].value = a[i].value \* b[j].value;

k++;

}

}

}

}

return k;

}

int main() {

int x,y;

cout<<"Enter number of non zero elements in matrix 1: ";

cin>>x;

triplet t[100];

cout << "Enter row, col, value:\n";

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

cin >> t[i].row >> t[i].column >> t[i].value;

}

cout<<"Enter number of non zero elements in matrix 2: ";

cin>>y;

triplet s[100];

cout << "Enter row, col, value:\n";

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

cin >> s[i].row >> s[i].column >> s[i].value;

}

triplet a[100];

transpose(t,a,x);

cout<<"Transpose of first matrix is ";

print\_sparse(a,x);

triplet b[100];

transpose(s,b,y);

cout<<"Transpose of second matrix is ";

print\_sparse(b,y); // Addition

Triplet sum[200];

int sumSize = add\_sparse(a, x, b, y, sum);

cout << "\nAddition of matrices:\n";

print\_sparse(sum, sumSize);

// Multiplication

Triplet prod[200];

int prodSize = multiply\_sparse(a, x, 10, 10, b, y, 10, 10, prod); // assuming max 10x10 matrices

cout << "\nMultiplication of matrices:\n";

print\_sparse(prod, prodSize);

return 0;

}

**Q7.**

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n];

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

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

cin >> arr[i];

int count = 0;

// Count inversions

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

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

if(arr[i] > arr[j])

count++;

}

}

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

return 0;

}

**Q8.**

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n];

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

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

cin >> arr[i];

}

int count = 0;

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

bool isDuplicate = false;

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

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

isDuplicate = true;

break;

}

}

if (!isDuplicate) {

count++; // Increment count if element is distinct

}

}

cout << "Total number of distinct elements: " << count << endl;

return 0;

}