Assignment -2

Name: Sahil Mangla

Roll no: 1024030359



1.#include <iostream>

using namespace std;

int main() {

int n = 4;

// (a) Diagonal Matrix → needs only n elements

int diag[4] = {5, 8, 9, 12};

cout << "Diagonal Matrix:\n";

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

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

if(i==j) cout << diag[i] << " ";

else cout << 0 << " ";

}

cout << "\n";

}

// (b) Tri-diagonal Matrix → needs 3n-2 elements

int tri[10] = {2, 3, 4,

5, 6, 7, 8,

9, 10, 11};

cout << "\nTri-diagonal Matrix:\n";

int k=0;

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

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

if(i==j) cout << tri[n-1+i] << " ";

else if(i==j+1) cout << tri[i-1] << " ";

else if(i+1==j) cout << tri[2\*n-1+i] << " ";

else cout << 0 << " ";

}

cout << "\n";

}

// (c) Lower Triangular Matrix → n(n+1)/2 elements

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

cout << "\nLower Triangular Matrix:\n";

k=0;

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

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

if(i>=j) cout << lower[i\*(i+1)/2 + j] << " ";

else cout << 0 << " ";

}

cout << "\n";

}

// (d) Upper Triangular Matrix → n(n+1)/2 elements

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

cout << "\nUpper Triangular Matrix:\n";

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

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

if(i<=j){

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

cout << upper[index] << " ";

} else cout << 0 << " ";

}

cout << "\n";

}

// (e) Symmetric Matrix → store lower part only

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

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

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

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

if(i>=j) cout << sym[i\*(i+1)/2 + j] << " ";

else cout << sym[j\*(j+1)/2 + i] << " ";

}

cout << "\n";

}

return 0;

}

2.#include <iostream>

#include <vector>

using namespace std;

long long mergeAndCount(vector<int>& arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

vector<int> L(n1), R(n2);

for(int i=0;i<n1;i++) L[i] = arr[left+i];

for(int j=0;j<n2;j++) R[j] = arr[mid+1+j];

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

long long invCount = 0;

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

if(L[i] <= R[j]){

arr[k++] = L[i++];

} else {

arr[k++] = R[j++];

invCount += (n1 - i);

}

}

while(i<n1) arr[k++] = L[i++];

while(j<n2) arr[k++] = R[j++];

return invCount;

}

long long mergeSortAndCount(vector<int>& arr, int left, int right){

long long invCount = 0;

if(left < right){

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

invCount += mergeSortAndCount(arr, left, mid);

invCount += mergeSortAndCount(arr, mid+1, right);

invCount += mergeAndCount(arr, left, mid, right);

}

return invCount;

}

int main(){

vector<int> arr = {2, 4, 1, 3, 5};

int n = arr.size();

long long inversions = mergeSortAndCount(arr, 0, n-1);

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

return 0;

}

3.#include <iostream>

#include <vector>

using namespace std;

int binarySearch(vector<int>& arr, int target) {

int low = 0;

int high = arr.size() - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == target) {

return mid;

}

else if (arr[mid] < target) {

low = mid + 1;

}

else {

high = mid - 1;

}

}

return -1;

}

int main() {

int n, target;

cout << "Enter number of elements: ";

cin >> n;

vector<int> arr(n);

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

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

cin >> arr[i];

}

cout << "Enter target to search: ";

cin >> target;

int result = binarySearch(arr, target);

if (result != -1) {

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

} else {

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

}

return 0;

}

4.#include<iostream>

using namespace std;

int main(){

vector<int> arr = {64,34,25,12,22,11,90};

int n = arr.size();

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

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

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

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

}

}

}

cout<<"Sorted Array is: ";

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

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

}cout<<endl;

return 0;

}

5.#include <bits/stdc++.h>

using namespace std;

int main() {

vector<int> arr = {1, 2, 2, 3, 4, 4, 5};

sort(arr.begin(), arr.end());

int distinct = 1;

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

if(arr[i] != arr[i-1]) distinct++;

}

cout << "Total distinct elements = " << distinct << endl;

}

6.#include<iostream>

using namespace std;

int main(){

vector<int> arr = {1,2,3,5,6,7,8};

int n = arr.size();

int cnt = 1;

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

if(arr[i]!=cnt){

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

break;

}

cnt++;

}

return 0;

}

7.#include<iostream>

#include<vector>

using namespace std;

int main(){

vector<int> arr = {1,2,3,4,6,7,8};

int n = arr.size();

int low = 0, high = n - 1, ans = n;

while(low <= high){

int mid = low + (high - low) / 2;

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

ans = mid;

high = mid - 1;

}else{

low = mid + 1;

}

}

cout << "Missing Number is: " << ans + 1 << endl;

return 0;

}

8.#include <bits/stdc++.h>

using namespace std;

int main(){

string s1, s2;

string s3 = "abcd";

getline(cin, s1);

getline(cin, s2);

// Concatenate

cout << s1 + s2 << endl;

s1.append(s2);

// Reverse

int n = s3.length();

int start = 0;

while(start < n/2){

swap(s3[start], s3[n-start-1]);

start++;

}

cout << s3 << endl;

// Remove Vowels

string res = "";

for(char c : s3){

if(c!='a' && c!='e' && c!='i' && c!='o' && c!='u' &&

c!='A' && c!='E' && c!='I' && c!='O' && c!='U'){

res += c;

}

}

cout << res << endl;

// Convert to lowercase

for(int i=0; i<s1.length(); i++){

s1[i] = tolower(s1[i]);

}

cout << s1 << endl;

//Upper

for(int i=0; i<s1.length(); i++){

s1[i] = toupper(s1[i]);

}

cout << s1 << endl;

//ALPHABATICAL ORDER

sort(s1.begin(),s1.end());

cout<<s1<<endl;

return 0;

}