|  |  |
| --- | --- |
| Assignment 2 Solutions | Ishaan mishra    1024030346 |

# Question 1

#include <iostream>

using namespace std;

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

int low = 0, high = n - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == key) return mid;

else if (arr[mid] < key) low = mid + 1;

else high = mid - 1;

}

return -1;

}

int main() {

int arr[] = {11, 12, 22, 25, 34, 64, 90};

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

int key;

cout << "Enter element to search: ";

cin >> key;

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

if (result != -1) cout << "Element found at position " << result + 1 << endl;

else cout << "Element not found\n";

return 0;

}

# Question 2

#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]) {

int temp = arr[j];

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

arr[j+1] = temp;

}

}

}

}

int main() {

int arr[] = {64, 34, 25, 12, 22, 11, 90};

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

cout << "Unsorted array: ";

for (int i = 0; i < n; i++) cout << arr[i] << " ";

cout << endl;

bubbleSort(arr, n);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) cout << arr[i] << " ";

cout << endl;

return 0;

}

# Question 3

// Question 3: Find Missing Number in a Sorted Array

#include <iostream>

using namespace std;

int func1(int arr[], int n) {

int total = (n + 1) \* (n + 2) / 2; // sum of 1..n+1

int sum = 0;

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

return total - sum;

}

int func2(int arr[], int n) {

int low = 0, high = n - 1;

while (low <= high) {

int mid = (low + high) / 2;

if (arr[mid] == mid + 1) low = mid + 1;

else high = mid - 1;

}

return low + 1;

}

int main(){

int n,i;

cout<<"Enter the number of elements in array";

cin >> n;

int arr[n];

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

cout << "\narr[" << i << "]";

cin >> arr[i];

}

int missing\_num = func1(arr,n);

cout << "\nmissing number found in linear time : " << missing\_num;

missing\_num = func2(arr,n);

cout << "\nmissing number found in logarithmic time : " << missing\_num;

}

# Question 4

// Question 4: String Related Programs

#include <iostream>

#include <string>

using namespace std;

int main(){

char dest[50] = "This is an";

char src[50] = " example";

strcat(dest, src);

cout << dest ;

return 0;

}

int main(){

string str, result = "";

cout << "Enter a string: ";

getline(cin, str);

for (char c : str) {

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

c=='A'||c=='E'||c=='I'||c=='O'||c=='U')) result += c;

}

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

return 0;

}

int main() {

char ch;

cout << "Enter an uppercase character: ";

cin >> ch;

if (ch >= 'A' && ch <= 'Z') {

ch = ch + 32; // ASCII difference

}

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

return 0;

}

int main(){

string str;

cout << "Enter a string: ";

getline(cin, str);

int n = str.length();

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

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

if (str[j] < str[i]) {

char temp = str[i];

str[i] = str[j];

str[j] = temp;

}

}

}

cout << "String sorted in alphabetical order: " << str << endl;

return 0;

}

# Question 5

// Question 5: Efficient Storage of Special Matrices  
/\*  
Diagonal Matrix – store only n diagonal elements in 1D array.  
Tri-diagonal Matrix – store 3n-2 elements.  
Lower Triangular Matrix – store n(n+1)/2 elements.  
Upper Triangular Matrix – store n(n+1)/2 elements.  
Symmetric Matrix – store n(n+1)/2 elements (A[i][j] = A[j][i]).  
Implementation depends on index mapping formulas.  
\*/

# Question 6

// Question 6: Sparse Matrix Operations using Triplet Representation  
#include <iostream>

using namespace std;

void printTriplet(int a[][3]) {

int n = a[0][2];

cout << "Row Col Val\n";

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

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

}

}

void transpose(int a[][3], int b[][3]) {

int n = a[0][2];

b[0][0] = a[0][1];

b[0][1] = a[0][0];

b[0][2] = n;

int k = 1;

for (int col = 0; col < a[0][1]; col++) {

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

if (a[i][1] == col) {

b[k][0] = a[i][1];

b[k][1] = a[i][0];

b[k][2] = a[i][2];

k++;

}

}

}

}

void add(int a[][3], int b[][3], int c[][3]) {

if (a[0][0] != b[0][0] || a[0][1] != b[0][1]) {

cout << "Addition not possible\n";

return;

}

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

c[0][0] = a[0][0];

c[0][1] = a[0][1];

while (i <= a[0][2] && j <= b[0][2]) {

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 <= a[0][2]) {

c[k][0] = a[i][0];

c[k][1] = a[i][1];

c[k][2] = a[i][2];

i++; k++;

}

while (j <= b[0][2]) {

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;

}

void multiply(int a[][3], int b[][3], int c[][3]) {

if (a[0][1] != b[0][0]) {

cout << "Multiplication not possible\n";

return;

}

int bT[20][3];

transpose(b, bT);

int pa = a[0][2], pb = bT[0][2];

int k = 1;

c[0][0] = a[0][0];

c[0][1] = b[0][1];

int i = 1;

while (i <= pa) {

int row = a[i][0];

int j = 1;

while (j <= pb) {

int col = bT[j][0];

int sum = 0;

int ii = i, jj = j;

while (ii <= pa && a[ii][0] == row && jj <= pb && bT[jj][0] == col) {

if (a[ii][1] < bT[jj][1]) ii++;

else if (a[ii][1] > bT[jj][1]) jj++;

else {

sum += a[ii][2] \* bT[jj][2];

ii++; jj++;

}

}

if (sum != 0) {

c[k][0] = row;

c[k][1] = col;

c[k][2] = sum;

k++;

}

while (j <= pb && bT[j][0] == col) j++;

}

while (i <= pa && a[i][0] == row) i++;

}

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

}

int main() {

int A[20][3] = {

{4, 4, 4},

{0, 0, 5},

{1, 1, 8},

{2, 3, 3},

{3, 0, 6}

};

int B[20][3] = {

{4, 4, 3},

{0, 0, 4},

{1, 2, 7},

{2, 3, 9}

};

cout << "Matrix A:\n"; printTriplet(A);

cout << "\nMatrix B:\n"; printTriplet(B);

int T[20][3], SUM[20][3], MUL[20][3];

transpose(A, T);

cout << "\nTranspose of A:\n"; printTriplet(T);

add(A, B, SUM);

cout << "\nA + B:\n"; printTriplet(SUM);

multiply(A, B, MUL);

cout << "\nA \* B:\n"; printTriplet(MUL);

return 0;

}

# Question 7

// Question 7: Count Inversions in Array

#include <iostream>

using namespace std;

int main() {

int n;

cout << "Enter number of elements: ";

cin >> n;

int arr[n];

cout << "Enter elements:\n";

for (int i=0; i<n; i++) cin >> arr[i];

int result=0;

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

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

if(arr[i]>arr[j]) result++;

}

}

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

return 0;

}

# Question 8

// Question 8: Count Distinct Elements in Array

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n];

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

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 distinct elements = " << distinctCount << endl;

return 0;

}