**Data Structure Lab Assignment -2**

**Question 1:** **Implement the binary search algorithm, regarded as a fast search algorithm with run-time complexity of Ο(log n) in comparison to the Linear Search.**

**Answer:**

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

**using namespace std;**

**int main()**

**{**

**int n = 7;**

**int arr[n]={3,5,7,8,9,12,15};**

**int num;**

**cout << "enter the number to be found in given array : ";**

**cin >> num;**

**cout << endl;**

**int low=0;**

**int high=n-1;**

**while(low<=high)**

**{**

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

**if(num==arr[mid])**

**{**

**cout << "number found at index :"<< mid << endl;**

**break;**

**}**

**else if (num<arr[mid])**

**{**

**high=mid-1;**

**}**

**else**

**{**

**low=mid+1;**

**}**

**}**

**return 0;**

**}**

**Question 2: Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. Code the Bubble sort with the following elements: 64 34 25 12 22 11 90.**

**Answer:**

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

**Using namespace std;**

**Int main ()**

**{**

**Int n=7;**

**Int arr[n] = {64,34,25,12,22,11,90};**

**For (int i =0 ;i<n; i++)**

**{**

**For (int j=0; j<n-i-1; j++)**

**{**

**If(arr[j]>arr[j+1])**

**{**

**Int temp;**

**Temp = arr[j];**

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

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

**}**

**}**

**}**

**Cout << “array after bubble sorting : “ << endl;**

**For (int i=0 ;i<n ; i++)**

**{**

**Cout << arr[i] << “ “ << endl;**

**}**

**Return 0;**

**}**

**Question 3:** **Design the Logic to Find a Missing Number in a Sorted Array. Given an array of n-1 distinct integers in the range of 1 to n, find the missing number in it in a Sorted Array (a) Linear time (b) Using binary search.**

**Answer:**

**linear time**

**#include <bits/stdC++.h>**

**using namespace std;**

**int main()**

**{**

**int n=7;**

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

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

**{**

**if(arr[i] != i+1)**

**{**

**cout << "missing number is : " << i+1 << endl;**

**break;**

**}**

**}**

**return 0;**

**}**

**Answer: using binary search**

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

**using namespace std;**

**int main()**

**{**

**int n=7;**

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

**int low =0;**

**int high = n-1;**

**while(low<=high)**

**{**

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

**if(arr[mid]==mid+1)**

**{**

**low =mid+1;**

**}**

**else**

**{**

**high=mid-1;**

**}**

**}**

**cout << "missing number is : " << low+1 << endl;**

**return 0;**

**}**

**Question 4: String Related Programs**

**(a) Write a program to concatenate one string to another string.**

**(b) Write a program to reverse a string.**

**(c) Write a program to delete all the vowels from the string.**

**(d) Write a program to sort the strings in alphabetical order.**

**(e) Write a program to convert a character from uppercase to lowercase.**

**Answer (a)**

**#include <iostream>**

**using namespace std;**

**int main() {**

**char str1[50] = "Hello ";**

**char str2[20] = "World";**

**int i = 0, j = 0;**

**while (str1[i] != '\0') {**

**i++;**

**}**

**while (str2[j] != '\0') {**

**str1[i] = str2[j];**

**i++;**

**j++;**

**}**

**str1[i] = '\0';**

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

**return 0;**

**}**

**Answer (b)**

**#include <iostream>**

**using namespace std;**

**int main() {**

**char str[50] = "Computer";**

**int len = 0;**

**while (str[len] != '\0') {**

**len++;**

**}**

**cout << "Reversed string: ";**

**for (int i = len-1; i >= 0; i--) {**

**cout << str[i];**

**}**

**return 0;**

**}**

**Answer (c)**

**#include <iostream>**

**using namespace std;**

**int main() {**

**char str[50] = "Hello World";**

**char result[50];**

**int i = 0, j = 0;**

**while (str[i] != '\0') {**

**char c = str[i];**

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

**c=='A'||c=='E'||c=='I'||c=='O'||c=='U')) {**

**result[j++] = c;**

**}**

**i++;**

**}**

**result[j] = '\0';**

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

**return 0;**

**}**

**Answer (d)**

**#include <iostream>**

**using namespace std;**

**int main() {**

**char str[5][20] = {"banana", "apple", "mango", "cherry", "pear"};**

**char temp[20];**

**for (int i = 0; i < 4; i++) {**

**for (int j = 0; j < 4 - i; j++) {**

**int k = 0;**

**while (str[j][k] != '\0' && str[j+1][k] != '\0' && str[j][k] == str[j+1][k]) {**

**k++;**

**}**

**if (str[j][k] > str[j+1][k]) {**

**int p = 0;**

**while (true) {**

**temp[p] = str[j][p];**

**str[j][p] = str[j+1][p];**

**str[j+1][p] = temp[p];**

**if (str[j][p] == '\0' && str[j+1][p] == '\0')**

**break;**

**p++;**

**}**

**}**

**}**

**}**

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

**for (int i = 0; i < 5; i++) {**

**cout << str[i] << endl;**

**}**

**return 0;**

**}**

**Answer (e)**

**#include <iostream>**

**using namespace std;**

**int main() {**

**char ch = 'A';**

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

**ch = ch + 32;**

**}**

**cout << "Lowercase: " << ch;**

**return 0;**

**}**

**Question 5:** **Space required to store any two-dimensional array is 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜ƒ 𝑟𝑜𝑤𝑠 × 𝑛𝑢𝑚𝑏𝑒𝑟 𝑜ƒ � �𝑜𝑙𝑢𝑚𝑛𝑠. Assuming an array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.**

**(a) Diagonal Matrix.**

**(b) Tri-diagonal Matrix.**

**(c) Lower triangular Matrix.**

**(d) Upper triangular Matrix.**

**(e) Symmetric Matrix.**

**Code: A**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n = 4;**

**int dia[4];**

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

**cin >> dia[i];**

**}**

**cout << "Diagonal Matrix:\n";**

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

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

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

**else cout << 0 << " ";**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**Code: B**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n;**

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

**cin >> n;**

**int size = 3 \* n - 2;**

**int tri[size];**

**cout << "Enter elements of tridiagonal matrix:\n";**

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

**cin >> tri[i];**

**}**

**cout << "\nTridiagonal Matrix:\n";**

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

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

**if (i == j) {**

**cout << tri[i] << " ";**

**} else if (i == j + 1) {**

**cout << tri[n + j] << " ";**

**} else if (i + 1 == j) {**

**cout << tri[2 \* n - 1 + i] << " ";**

**} else {**

**cout << 0 << " ";**

**}**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**Code : C**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n;**

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

**cin >> n;**

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

**int lower[size];**

**cout << "Enter elements of lower triangular matrix:\n";**

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

**cin >> lower[i];**

**}**

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

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

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

**if (i >= j) {**

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

**cout << lower[index] << " ";**

**} else {**

**cout << 0 << " ";**

**}**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**Code : D**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n;**

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

**cin >> n;**

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

**int upper[size];**

**cout << "Enter elements of upper triangular matrix:\n";**

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

**cin >> upper[i];**

**}**

**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 << endl;**

**}**

**return 0;**

**}**

**Code : E**

**#include <iostream>**

**using namespace std;**

**int main() {**

**int n;**

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

**cin >> n;**

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

**int sym[size];**

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

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

**cin >> sym[i];**

**}**

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

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

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

**if (i >= j) {**

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

**cout << sym[index] << " ";**

**} else {**

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

**cout << sym[index] << " ";**

**}**

**}**

**cout << endl;**

**}**

**return 0;**

**}**

**Question 6:** **Write a program to implement the following operations on a Sparse Matrix, assuming the matrix is represented using a triplet.**

**(a) Transpose of a matrix.**

**(b) Addition of two matrices.**

**(c) Multiplication of two matrices.**

**Code : A**

**#include <iostream>**

**using namespace std;**

**struct Triplet {**

**int row, col, val;**

**};**

**int main() {**

**Triplet A[] = {{0, 0, 5}, {0, 2, 8}, {1, 1, 3}};**

**int n = 3;**

**cout << "Transpose of Sparse Matrix:\n";**

**for (int c = 0; c < 3; c++) {**

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

**if (A[i].col == c) {**

**cout << A[i].col << " " << A[i].row << " " << A[i].val << endl;**

**}**

**}**

**}**

**return 0;**

**}**

**Code :** B

#include <iostream>

using namespace std;

struct Triplet {

int row, col, val;

};

int main() {

Triplet A[] = {{0, 0, 5}, {0, 2, 8}, {1, 1, 3}};

Triplet B[] = {{0, 0, 2}, {1, 1, 4}, {2, 2, 7}};

int n1 = 3, n2 = 3;

cout << "Addition of Sparse Matrices:\n";

int i = 0, j = 0;

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

if (A[i].row == B[j].row && A[i].col == B[j].col) {

cout << A[i].row << " " << A[i].col << " " << A[i].val + B[j].val << endl;

i++; j++;

} else if (A[i].row < B[j].row || (A[i].row == B[j].row && A[i].col < B[j].col)) {

cout << A[i].row << " " << A[i].col << " " << A[i].val << endl;

i++;

} else {

cout << B[j].row << " " << B[j].col << " " << B[j].val << endl;

j++;

}

}

while (i < n1) {

cout << A[i].row << " " << A[i].col << " " << A[i].val << endl;

i++;

}

while (j < n2) {

cout << B[j].row << " " << B[j].col << " " << B[j].val << endl;

j++;

}

return 0;

}

Code :C

**#include <iostream>**

**using namespace std;**

**struct Triplet {**

**int row, col, val;**

**};**

**int main() {**

**Triplet A[] = {{0, 0, 5}, {0, 2, 8}, {1, 1, 3}};**

**Triplet B[] = {{0, 0, 2}, {1, 1, 4}, {2, 2, 7}};**

**int n1 = 3, n2 = 3;**

**int result[10][10] = {0}; // assuming max size 10x10**

**for (int x = 0; x < n1; x++) {**

**for (int y = 0; y < n2; y++) {**

**if (A[x].col == B[y].row) {**

**result[A[x].row][B[y].col] += A[x].val \* B[y].val;**

**}**

**}**

**}**

**cout << "Multiplication of Sparse Matrices:\n";**

**for (int i = 0; i < 3; i++) {**

**for (int j = 0; j < 3; j++) {**

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

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

**}**

**}**

**}**

**return 0;**

**}**

**Question 7:** **Let A[1 …. n] be an array of n real numbers. A pair (A[i], A[j ]) is said to be an inversion if these numbers are out of order, i.e., i < j but A[i]>A[j ]. Write a program to count the number of inversions in an array.**

**Code :**

**Question 8:** **Write a program to count the total number of distinct elements in an array of length n.**

**Code :**

**#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;**

**return 0;**

**}**