## DATA STRUCTURES AND ALGORITHMS

## ASSIGNMENT- 2

Name-Harsh

Roll\_no-1024030497

Batch-2C25

Ques-1)

#include<iostream>

using namespace std;

int main()

{

int a[5]={1,2,3,4,5};

int n=sizeof(a)/sizeof(int);

int start=0;

int end=n-1;

int x;

cout<<"element you wanna search"<<endl;

cin>>x;

bool check=false;

while(start<=end)

{

int mid=(start+end)/2;

if(a[mid]==x)

{

check=true;

cout<<"element fount at index"<<mid<<endl;

break;

}

else if(a[mid]<x) start=mid+1;

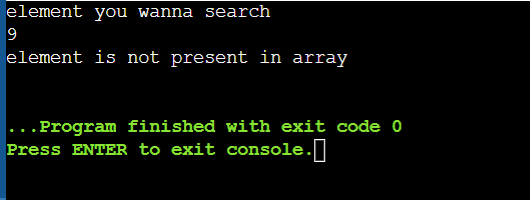
else if(a[mid]>x) end=mid-1;

}

if(check==false) cout<<"element is not present in array"<<endl;

return 0;

}



Ques-2)

#include<iostream>

using namespace std;

int main()

{

int a[7]={64,34,25,11,12,22,90};

int n=7;

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

{

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

{

if(a[j]<a[j-1])

{

int temp=a[j];

a[j]=a[j-1];

a[j-1]=temp;

}

}

}

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

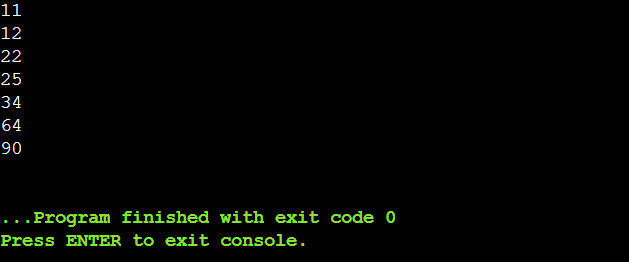
{

cout<<a[k]<<endl;

}

return 0;

}



Ques-3)

#include<iostream>

using namespace std;

int main()

{

int a[5]={1,2,3,4,6};

int n=sizeof(a)/sizeof(int);

int sum=0;

int total\_sum;

total\_sum=((n+1)\*(n+2)/2);

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

{

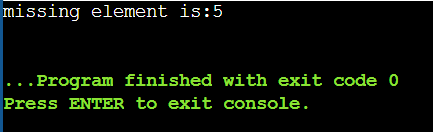
sum=sum+a[i];

}

cout<<"missing element is:"<<total\_sum-sum<<endl;

return 0;

}



Binary search approach-

#include <iostream>

using namespace std;

int main() {

int n;

cin >> n;

int arr[100];

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

cin >> arr[i];

}

int low = 0, high = n - 2;

int missing;

while (low <= high) {

int mid = (low + high) / 2;

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

low = mid + 1;

} else {

high = mid - 1;

}

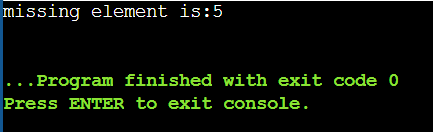
}

missing = low + 1;

cout << missing << endl;

return 0;

}



Ques-4)

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char str1[100], str2[100];

cout << "Enter first string: ";

cin >> str1;

cout << "Enter second string: ";

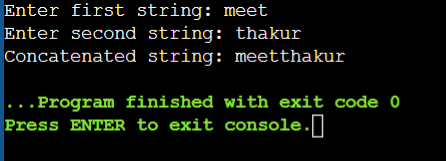
cin >> str2;

strcat(str1, str2);

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

return 0;

}



To reverse a string

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char str[100];

cout << "Enter a string: ";

cin >> str;

int n = strlen(str);

cout << "Reversed string: ";

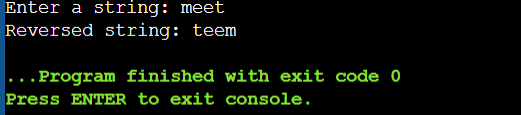
for (int i = n - 1; i >= 0; i--) {

cout << str[i];

}

return 0;

}



To delete all the vowels from string-

#include <iostream>

#include <cstring>

using namespace std;

int main() {

char str[100];

cout << "Enter a string: ";

cin >> str;

char result[100];

int j = 0;

for (int i = 0; i < strlen(str); 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[j] = str[i];

j = j + 1;

}

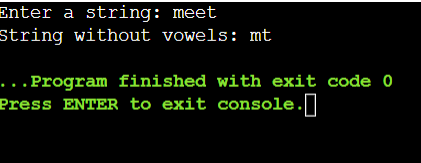
}

result[j] = '\0';

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

return 0;

}



To sort the elements in alphabetical order

#include <iostream>

#include <cstring>

using namespace std;

int main() {

int n;

cout << "Enter number of strings: ";

cin >> n;

char str[50][50], temp[50];

cout << "Enter strings:\n";

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

cin >> str[i];

}

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

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

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

strcpy(temp, str[i]);

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

strcpy(str[j], temp);

}

}

}

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

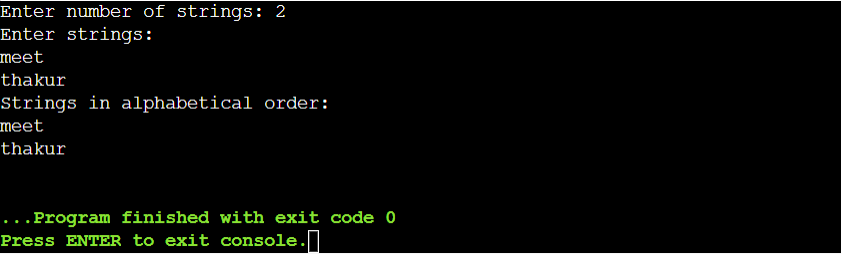
for (int i = 0; i < n; i++) {

cout << str[i] << endl;

}

return 0;

}



To convert uppercase to lowercase-

#include <iostream>

using namespace std;

int main() {

char ch;

cout << "Enter an uppercase character: ";

cin >> ch;

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

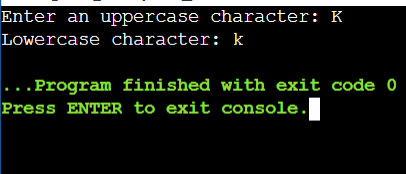
ch = ch + 32;

}

cout << "Lowercase character: " << ch;

return 0;

}



Ques-5)

Diagonal elements-

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n]; // Store only diagonal elements

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

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

cin >> arr[i];

}

cout << "Matrix is:\n";

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

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

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

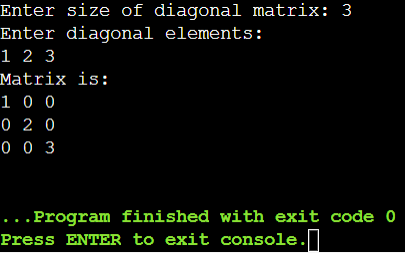
else cout << "0 ";

}

cout << endl;

}

}



b) Lower triangular matrix-

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n\*(n+1)/2];

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

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

int k = 0;

cout << "Matrix is:\n";

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

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

if(j <= i) cout << arr[k++] << " ";

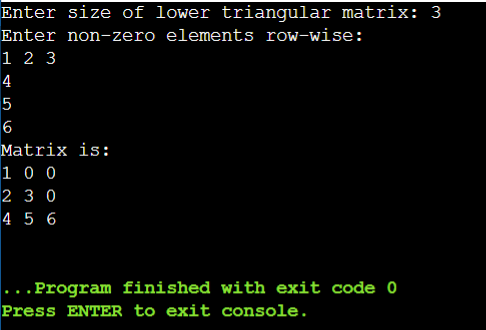
else cout << "0 ";

}

cout << endl;

}

}



Upper triangular matrix-

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n\*(n+1)/2];

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

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

int k = 0;

cout << "Matrix is:\n";

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

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

if(j >= i) cout << arr[k++] << " ";

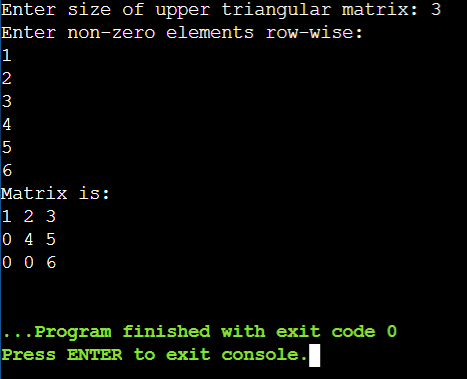
else cout << "0 ";

}

cout << endl;

}

}



Symmetrical matrix-

#include <iostream>

using namespace std;

int main() {

int n;

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

cin >> n;

int arr[n\*(n+1)/2];

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

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

cout << "Matrix is:\n";

int k = 0;

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

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

if(j <= i) cout << arr[k++] << " ";

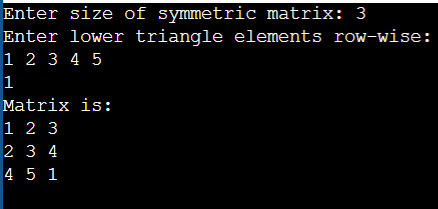
else cout << arr[i\*(i+1)/2 + j] << " "; // fetch symmetric

}

cout << endl;

}

}



Ques-6)

#include <iostream>

using namespace std;

int main() {

int A[100][3], T[100][3];

int rows, cols, nonZero;

int i, j, k = 0;

cout << "Enter rows, cols, and number of non-zero elements: ";

cin >> rows >> cols >> nonZero;

cout << "Enter triplets (row col value):\n";

for (i = 0; i < nonZero; i++)

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

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

for (j = 0; j < nonZero; j++) {

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

T[k][0] = A[j][1];

T[k][1] = A[j][0];

T[k][2] = A[j][2];

k++;

}

}

}

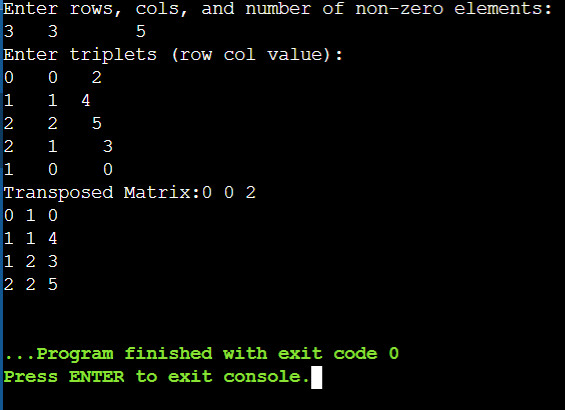
cout << "Transposed Matrix:";

for (i = 0; i < k; i++)

cout << T[i][0] << " " << T[i][1] << " " << T[i][2] << "\n";

return 0;

}



Addition of sparx matrix-

#include <iostream>

using namespace std;

int main() {

int A[100][3], B[100][3], SUM[200][3];

int m1, n1, t1, m2, n2, t2;

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

cout << "Enter rows, cols, and non-zero count for Matrix A: ";

cin >> m1 >> n1 >> t1;

cout << "Enter triplets for Matrix A:\n";

for (int x = 0; x < t1; x++)

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

cout << "Enter rows, cols, and non-zero count for Matrix B: ";

cin >> m2 >> n2 >> t2;

cout << "Enter triplets for Matrix B:\n";

for (int x = 0; x < t2; x++)

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

if (m1 != m2 || n1 != n2) {

cout << "Matrix dimensions mismatch!\n";

return 0;

}

while (i < t1 && j < t2) {

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

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

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

SUM[k][2] = A[i][2] + B[j][2];

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

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

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

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

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

i++; k++;

} else {

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

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

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

j++; k++;

}

}

while (i < t1) {

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

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

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

i++; k++;

}

while (j < t2) {

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

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

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

j++; k++;

}

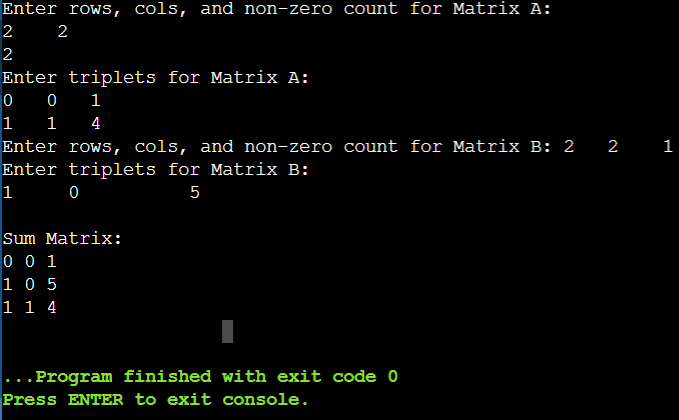
cout << "\nSum Matrix:\n";

for (i = 0; i < k; i++)

cout << SUM[i][0] << " " << SUM[i][1] << " " << SUM[i][2] << "\n";

return 0;

}



c)multiplication of sparx matrix-

#include <iostream>

using namespace std;

int main() {

int A[100][3], B[100][3], PROD[100][3];

int m1, n1, t1, m2, n2, t2;

int i, j, k, p = 0;

cout << "Enter rows, cols, and non-zero count for Matrix A: ";

cin >> m1 >> n1 >> t1;

cout << "Enter triplets for Matrix A:\n";

for (i = 0; i < t1; i++)

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

cout << "Enter rows, cols, and non-zero count for Matrix B: ";

cin >> m2 >> n2 >> t2;

cout << "Enter triplets for Matrix B:\n";

for (i = 0; i < t2; i++)

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

if (n1 != m2) {

cout << "Matrix dimensions mismatch!\n";

return 0;

}

for (i = 0; i < m1; i++) {

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

int sum = 0;

for (k = 0; k < t1; k++) {

if (A[k][0] == i) {

for (int l = 0; l < t2; l++) {

if (B[l][0] == A[k][1] && B[l][1] == j)

sum += A[k][2] \* B[l][2];

}

}

}

if (sum != 0) {

PROD[p][0] = i;

PROD[p][1] = j;

PROD[p][2] = sum;

p++;

}

}

}

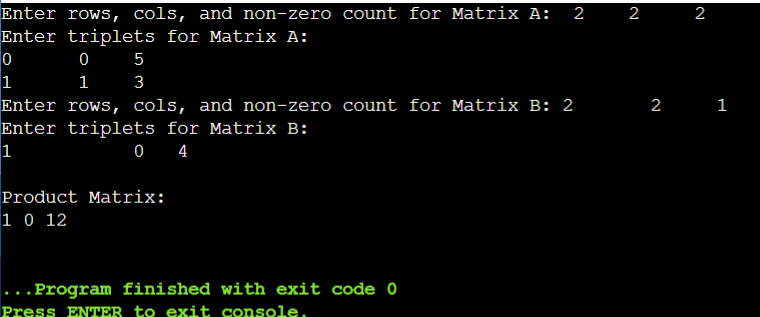
cout << "\nProduct Matrix:\n";

for (i = 0; i < p; i++)

cout << PROD[i][0] << " " << PROD[i][1] << " " << PROD[i][2] << "\n";

return 0;

}



Ques-7)

#include <iostream>

using namespace std;

int main() {

int A[100], n, i, j, count = 0;

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

cin >> n;

cout << "Enter " << n << " real numbers:\n";

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

cin >> A[i];

// Count inversions: i < j and A[i] > A[j]

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

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

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

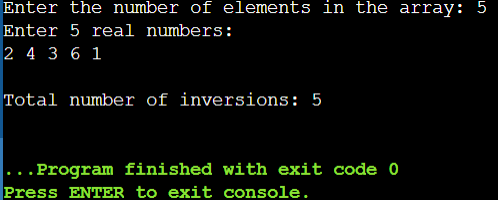
count++;

}

}

cout << "\nTotal number of inversions: " << count << "\n";

return 0;

}

Ques-8)

#include <iostream>

using namespace std;

int main() {

int A[100], n, i, j, count = 0;

bool isDistinct;

cout << "Enter number of elements: ";

cin >> n;

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

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

cin >> A[i];

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

isDistinct = true;

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

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

isDistinct = false;

break;

}

}

if (isDistinct)

count++;

}

cout << "\nTotal distinct elements: " << count << "\n";

return 0;

}

