**PSG COLLEGE OF TECHNOLOGY, COIMBATORE -641004**

**DEPARTMENT OF COMPUTER APPLICATIONS**

**I SEMESTER MCA**

**23MX16 C PROGRAMMING LABORATORY**

**PROBLEM SHEET 4 – ARRAYS**

1. Consider an array of numbers from 1 to N . In this array, one of the numbers gets duplicated and one is missing. Write a C program to find out the duplicated number. Condition: Using only one loop and without any extra memory.

CODE:

#include <stdio.h>

int main() {

int n,flag=0;

printf("Enter no of elements:");

scanf("%d",&n);

int arr[n];

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

printf("Enter elements[%d]=",i);

scanf("%d",&arr[i]);

}

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

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

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

flag=1;

printf("Duplicate:%d\n",arr[i]);

}

}

}

if(flag==0){

printf("No duplicate");

}

return 0;

2. Write a program for printing the elements of a two-dimensional array (not necessarily square) in each of the following orders:

a. To-and-fro row-major order.

#include <stdio.h>

int main() {

int m,n;

printf("Enter no of rows:");

scanf("%d",&m);

printf("Enter no of cols:");

scanf("%d",&n);

int arr[m][n];

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

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

printf("Enter elements[%d][%d]=",i,j);

scanf("%d",&arr[i][j]);

}

}

//To and fro row

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

if(i%2==0){

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

printf("%d ",arr[i][j]);

}

}

else{

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

printf("%d ",arr[i][j]);

}

}

}

return 0;

}

B. Diagonal-major order.

CODE:

#include<stdio.h>

void main(){

    int n,m;

    printf("Enter the size of an 2d array arr[m][n](m,n):");

    scanf("%d,%d",&m,&n);

    int arr[m][n];

    for(int i=0;i<m;i++){

        for(int j=0;j<n;j++){

            printf("Enter the value of arr[%d][%d]:",i,j);

            scanf("%d",&arr[i][j]);

        }

    }

    //Diagonal row major order

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

        int i = 0, j = col;

        while (i < m && j >= 0) {

            printf("%d ", arr[i][j]);

            i++; j--;

        }

    }

    for (int row = 1; row < m; row++) {

        int i = row, j = n - 1;

        while (i < m && j >= 0) {

            printf("%d ", arr[i][j]);

            i++; j--;

        }

    }

}

c. Spiral order.

CODE:

#include<stdio.h>

void main(){

    int n,m;

    printf("Enter the size of an 2d array arr[m][n](m,n):");

    scanf("%d,%d",&m,&n);

    int arr[m][n];

    for(int i=0;i<m;i++){

        for(int j=0;j<n;j++){

            printf("Enter the value of arr[%d][%d]:",i,j);

            scanf("%d",&arr[i][j]);

        }

    }

    int top=0,bottom=m-1;

    int left=0,right=n-1;

    while (top<=bottom && left<=right){

        for (int j=left;j<=right;j++){

            printf("%d ", arr[top][j]);}

        top++;

        for (int i = top; i <= bottom; i++){

            printf("%d ", arr[i][right]);}

        right--;

        if (top<=bottom){

            for (int j=right;j>=left;j--){

                printf("%d ", arr[bottom][j]);}

            bottom--;

        }

        if (left<=right){

            for (int i=bottom;i>=top;i--){

                printf("%d ", arr[i][left]);}

            left++;

        }

    }

}

Notice that the diagonal-major order makes enough sense for square matrices. For general mxn matrices, take the length of each diagonal to be m and treat the elements as organized in a wrap-around fashion. For example, consider the 4x5 matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 |

The listing of its elements in the to-and-fro row-major order is:

1 2 3 4 5 10 9 8 7 6 11 12 13 14 15 20 19 18 17 16

The listing of the elements in the diagonal-major order is:

1 7 13 19 2 8 14 20 3 9 15 4 10 5 1611 17 5 6 12 18

The listing of the elements in the spiral order is:

1. 2 3 4 5 10 15 20 19 18 17 16 11 6 7 8 9 14 13 12 7
2. Search an element in an array where difference between adjacent elements is 1. Given an array where difference between adjacent elements is 1, write C program search for an element in the array and return the position of the element (return the first occurrence).

CODE:

#include<stdio.h>

#include<stdlib.h>

int main()

{

int i=0,n=9;

int arr[9]={3,4,6,8,11,16,18,22,88};

int flag=0;

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

if(abs(arr[i]-arr[i+1])==1){

printf("Element found at position:%d\n",i);

flag=1;

}

}

if(flag=0){

printf("Elment not found!!\n");

}

return 0;

}

4. Find maximum average subarray of k length Given an array with positive and negative numbers, find the maximum average subarray of given length.

Input: arr[] = {1, 12, -5, -6, 50, 3}, k = 4

Output: Maximum average subarray of length 4 begins at index 1.

Maximum average is (12 - 5 - 6 + 50)/4 = 51/4

CODE:

#include<stdio.h>

#include<stdlib.h>

int main()

{

int arr[]={1,12,-5,-6,50,3};

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

int k,sum=0,avg;

printf("Enter the value of k:");

scanf("%d",&k);

if(k>n){

printf("k is larger than the array size.\n");

return 1;

}

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

sum += arr[i];

}

int max\_sum = sum;

int max\_start\_index = 0;

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

sum = sum - arr[i - k] + arr[i];

if (sum > max\_sum) {

max\_sum = sum;

max\_start\_index = i - k + 1;

}

}

double max\_avg = (double)max\_sum / k;

printf("Maximum average subarray of length %d begins at index %d.\n", k, max\_start\_index);

printf("Maximum average is %.2f\n", max\_avg);

return 0;

}

**Deadline: 10.09.2025**