# Q1. WAP to add elements of an array using pointers. Use printing of your choice.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int arr[5];

//HERE INITIALISATION IN ONE LINE

printf("Enter the value of the 5 elements inside the array -->\n");

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

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

}

int \*p=&arr[0];

int sum=0;

for(int j=0;j<5;++j){

p=&arr[j];

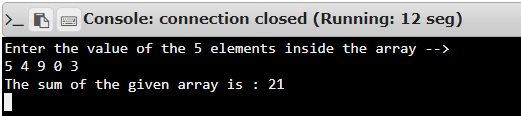
sum = sum + \*(p);

}

printf("The sum of the given array is : %d\n",sum);

return 3;

}



# Q2. Check if two same-sized arrays are equal or not. Print either 'equal' or 'unequal'.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int A[5];

int B[5];

int \*p=&A[0];

int \*q=&B[0];

printf("Enter the value of 5 elements inside array A -->\n");

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

p=&A[i];

scanf("%d",&\*(p));

//ALSO WRITTEN AS

//scanf("%d",&\*(p+i));

}

printf("Enter the value of 5 elements inside array B -->\n");

for(int j=0;j<5;++j){

q=&B[j];

scanf("%d",&\*(q));

}

//HERE COMPAIRING PROGRAM BEGINS

int count=0;

for(int k=0;k<5;++k){

p=&A[k];

q=&B[k];

if(\*p == \*q){

count=count+1;

continue;

}

else{

printf("unequal\n");

break;

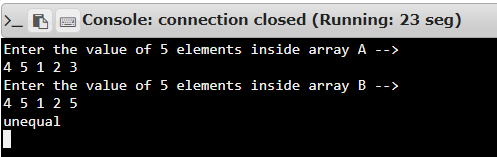
}

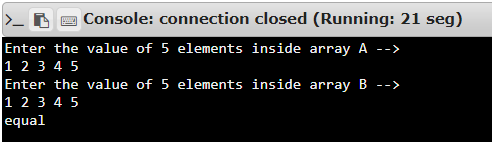
}

if(count==5) printf("equal\n");

return 3;

}





# Q3. WAP to swap two same-sized arrays using pointers.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n;

printf("Enter the size of the array --> ");

scanf("%d",&n);

int A[n];

int B[n];

int C[n];

//HERE C IS THE TEMPORARY ARRAY

int \*p=A,\*q=B,\*r=C;

printf("Enter the value of elements inside the array A -->\n");

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

printf("A[%d] is ",i);

scanf("%d",&\*(p+i));

}

printf("Enter the value of elements inside the array B -->\n");

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

printf("B[%d] is ",j);

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

\*(q+j)=B[j];

C[j]=B[j];

\*(r+j)=C[j];

}

//SWAPPING PROCESS BEGINS

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

\*(q+k)=\*(p+k);

\*(p+k)=\*(r+k);

}

//PRINTING SWAPPED VALUES

printf("\nAfter swapping\n");

printf("New values of array A -->\n");

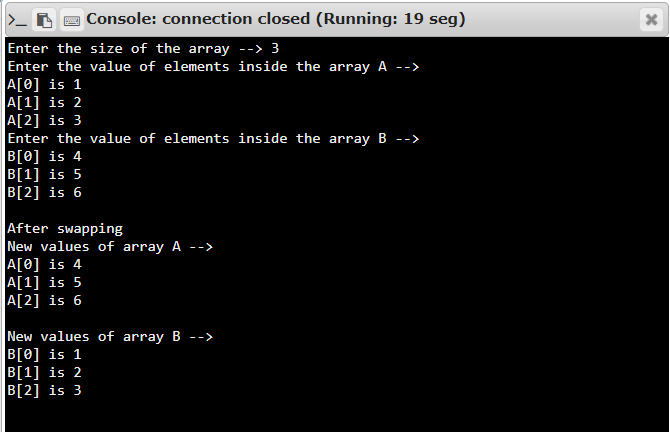
for(int i=0;i<n;++i) printf("A[%d] is %d\n",i,\*(p+i));

printf("\nNew values of array B -->\n");

for(int j=0;j<n;++j) printf("B[%d] is %d\n",j,\*(q+j));

return 3;

}



# Q4. WAP to reverse an array using pointers. Manipulate the original array itself instead of storing it in a new array.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n;

printf("Enter the size of the array --> ");

scanf("%d",&n);

int A[n];

int \*p=A;

printf("Enter the value of elements inside the array -->\n");

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

printf("A[%d] is : ",i);

scanf("%d",&\*(p+i));

}

int temp;

int start = 0;

int end = n-1;

while(start<=end){

temp = \*(p+start);

\*(p+start) = \*(p+end);

\*(p+end) = temp;

start++;

end--;

}

printf("Reversed value of the given array is -->\n");

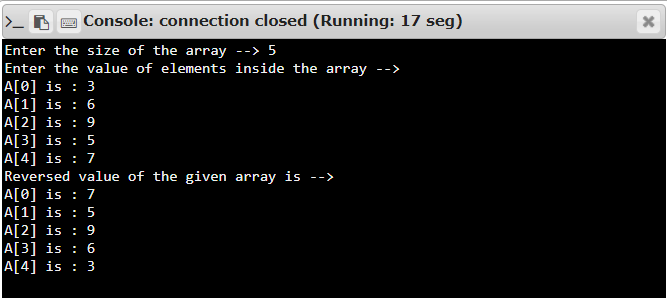
for(int i=0;i<n;++i){

printf("A[%d] is : %d\n",i,A[i]);

}

return 3;

}



# Q5. WAP to perform a linear search in an integer array using pointers.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n,i;

printf("Enter the size of the array -->\n");

scanf("%d",&n);

int A[n];

int \*p=A;

printf("\nEnter the value of the elements inside array A and last value as the search element -->\n");

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

printf("A[%d] is : ",i);

scanf("%d",&\*(p+i));

}

int count=0;

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

if(\*(p+i)==\*(p+n-1)){

printf("\n%d is the %dth element of the array A\n",\*(p+n-1),i+1);

break;

}

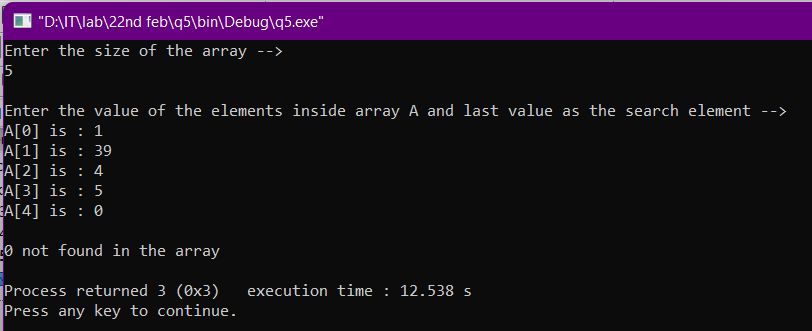
else ++count;

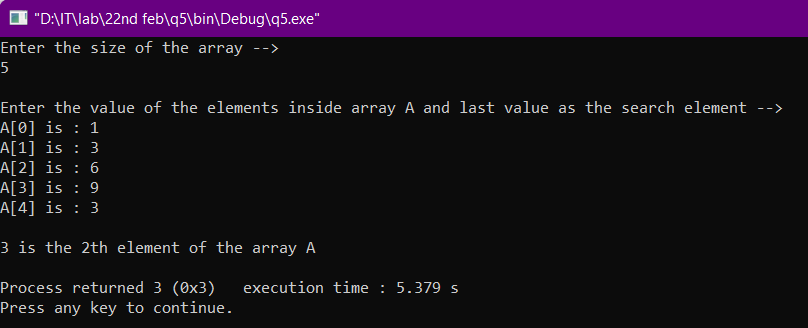
}

if(count==(n-1)) printf("\n%d not found in the array\n",\*(p+n-1));

return 3;

}





# Q6. WAP to use pointers for denoting a 2D array (matrix) and find the following:

# Mean value

# Max value

# Min value

# Median value (middle of all)

# Mode value (most frequent)

#include <stdio.h>

#include <stdlib.h>

int main()

{

int r,c,i;

printf("Enter the no of rows and columns of the arr : ");

scanf("%d %d",&r,&c);

int arr[r][c];

int (\*p)[c];

p=arr;

// int \*\*p;

float sum=0.0;

printf("\nEnter the value of the elements inside the array -->\n");

//ENTERING IN ROW MAJOR FORMAT

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

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

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

scanf("%d",&\*(\*(p+i)+j));

sum+=\*(\*(p+i)+j);

}

}

//PRINTING MEAN VALUE

printf("\nThe Mean Value of the given array arr is : %.3f\n",(sum/(r\*c)));

//FINDING MIN & MAX VALUES

int min=arr[0][0];

int max=arr[0][0];

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

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

if(\*(\*(p+i)+j)>=max){

max=\*(\*(p+i)+j);

}

if(\*(\*(p+i)+j)<=min) min=\*(\*(p+i)+j);

}

}

printf("The Max Value of the array arr is : %d\n",max);

printf("The Min Value of the array arr is : %d\n",min);

//FINDING THE MEDIAN VALUE(middle of all)

//FIRST SORTING THE ARRAY

int temp;

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

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

temp=\*(\*(p+i)+j);

if(\*(\*(p+i)+j)>\*(\*(p+i)+j+1)){

\*(\*(p+i)+j)=\*(\*(p+i)+j+1);

\*(\*(p+i)+j+1)=temp;

}

}

}

//INITIALIZED AN ARRAY A

int A[r\*c];

int k=0;

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

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

A[k]=\*(\*(p+i)+j);

++k;

}

}

//PRINTING MEDIAN VALUE

int median=0;

if((r\*c)%2==0){

median=(r\*c)/2;

printf("The Median Value of the array is : %.3f\n",(A[median-1]+A[median])/2.0);

}

else{

median=((r\*c)/2)+1;

printf("The Median Value of the array is : %d\n",A[median]);

}

//FOR MODE VALUE

int maxcount =1,count=1,res=A[0];

for(i=1;i<r\*c;++i){

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

++count;

}

else{

if(count>maxcount){

maxcount=count;

res=A[i-1];

}

count=1;

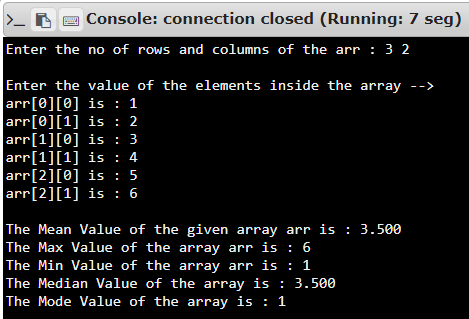
}

}

printf("The Mode Value of the array is : %d\n",res);

return 3;

}



# Q7. WAP to find the sum, difference, and product of two NxN matrices. Take matrix size from the user. Use pointers.

#include <stdio.h>

#include <stdlib.h>

int main()

{

int n;

printf("Enter the order 'N' for the two N\*N matrix : ");

scanf("%d",&n);

int A[n][n];

int B[n][n];

int (\*p)[n];

int (\*q)[n];

p=A;

q=B;

printf("\nEnter the values inside the matrix A -->\n");

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

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

printf("A[%d][%d] is : ",i,j);

scanf("%d",&\*(\*(p+i)+j));

}

}

printf("\nEnter the values inside the matrix B -->\n");

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

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

printf("B[%d][%d] is : ",i,j);

scanf("%d",&\*(\*(q+i)+j));

}

}

//ADDING AND DIFFERENCE OF TWO MATRIX

int C[n][n];

int D[n][n];

int (\*r)[n];

int (\*s)[n];

r=C;

s=D;

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

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

\*(\*(r+i)+j)= (\*(\*(p+i)+j)) + (\*(\*(q+i)+j));

\*(\*(s+i)+j)= (\*(\*(p+i)+j)) - (\*(\*(q+i)+j));

}

}

printf("\nSum of matrix A & B is -->\n");

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

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

printf("%d ",\*(\*(r+i)+j));

}

printf("\n");

}

printf("\nDifference of matrix A & B is -->\n");

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

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

printf("%d ",\*(\*(s+i)+j));

}

printf("\n");

}

//PRODUCT OF MATRIX A & B

int arr[n][n];

int (\*a)[n];

a=arr;

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

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

int sum=0;

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

sum+= (\*(\*(p+i)+k)) \* (\*(\*(q+k)+j));

}

\*(\*(a+i)+j)=sum;

}

}

printf("\nProduct of matrix A & B is -->\n");

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

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

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

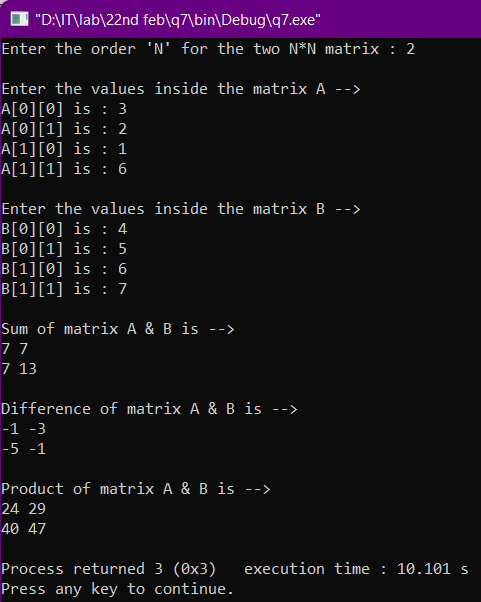
}

printf("\n");

}

return 3;

}



# Q8. WAP to use pointers to check if a matrix is:

# symmetric/asymmetric/skew-symmetric

# identity matrix

# invertible

#include <stdio.h>

#include <stdlib.h>

int main()

{

int A[3][3];

int i,j,k;

int (\*p)[3];

p=A;

printf("Enter the elements inside a 3\*3 matrix A -->\n");

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

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

scanf("%d",&\*(\*(p+i)+j));

}

}

//TRANSPOSE OF MATRIX A

int B[3][3];

int (\*q)[3];

q=B;

printf("\nTranspose of the matrix A is -->\n");

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

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

\*(\*(q+i)+j)=\*(\*(p+j)+i);

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

}

printf("\n");

}

//COMPAIRING BOTH A & A^t

int count=0;

int skew=0;

int identity=0;

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

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

if(\*(\*(p+i)+j)==\*(\*(q+i)+j)){

count+=1;

}

if(\*(\*(p+i)+j)==-(\*(\*(q+i)+j))){

skew+=1;

}

if(i==j){

if(\*(\*(p+i)+j)==1)

identity+=1;

}

}

}

int det = A[0][0] \* ((A[1][1]\*A[2][2]) - (A[2][1]\*A[1][2])) -A[0][1] \* (A[1][0]

\* A[2][2] - A[2][0] \* A[1][2]) + A[0][2] \* (A[1][0] \* A[2][1] - A[2][0] \* A[1][1]);

printf("\nDeterminant of matrix A is %d",det);

if(count==9) printf("\nThe given matrix A is symmetric.\n");

else if(skew==9) printf("\nThe given matrix A is skew-symmetric.\n");

else printf("\nThe given matrix A is asymmetric.\n");

if(identity==3) printf("The given matrix A is identity matrix.\n");

else printf("The given matrix A is not identity matrix.\n");

if(det!=0) printf("The given matrix A is invertible.\n");

else printf("The given matrix A is not invertible.\n");

return 3;

}

