**Multi-Dimensional Array related problems**

**(Total 15 questions)**

|  |  |  |
| --- | --- | --- |
| **SL** | **Problem statement** | **Difficulty levels** |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | WAP that will take 9 integers into a 3 by 3 array (2D) and show them as traditional matrix view.     |  |  | | --- | --- | | **Sample input** | **Sample output** | | 9 8 7 6 5 4 3 2 1 | 9 8 7  6 5 4  3 2 1 | | 1 1 1 2 2 2 3 3 3 | 1 1 1  2 2 2  3 3 3 |   #include <stdio.h>  int main()  {  int i,j;  int a[3][3];  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  scanf("%d",&a[i][j]);  }  }  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  printf("%d ",a[i][j]);  }  printf("\n");  }  return 0;  } | \* |
|  | WAP that will take (m x n) integers into a *m by n* array (2D) and print them both row-wise and column-wise.     |  |  | | --- | --- | | **Sample input (m,n)** | **Sample output** | | 2 3  1 2 3  6 5 4 | Row-wise: 1 2 3 6 5 4  Column-wise: 1 6 2 5 3 4 | | 3 3  1 1 1  2 2 2  3 3 3 | Row-wise: 1 1 1 2 2 2 3 3 3  Column-wise: 1 2 3 1 2 3 1 2 3 |   #include <stdio.h>  int main(){  int row,col,i,j;  scanf("%d %d",&row,&col);  int a[row][col];  printf("Enter the array of the matrix:\n ");  for(i=0;i<row;i++){  for(j=0;j<col;j++){  scanf("%d",&a[i][j]) ;  }  }  for(i=0;i<row;i++){  for(j=0;j<col;j++){  printf("%d ",a[i][j]) ;  }  printf("\n");  }  printf("Row-wise: \n");  for(i=0;i<row;i++){  for(j=0;j<col;j++){  printf("%d ",a[i][j]) ;  }  }  printf("\nColumn-wise: \n");  for(j=0;j<col;j++){  for(i=0;i<row;i++){  printf("%d ",a[i][j]) ;  }  }  return 0;  } | \* |
|  | WAP that will take inputs of a 3 by 3 matrix into a 2D array. Now find the determinant of this matrix. <http://www.mathsisfun.com/algebra/matrix-determinant.html>   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 1 2 3  4 5 6  7 8 9 | 0 |   #include <stdio.h>  int main()  {  int i,j,x,y,z,det;  int a[3][3];  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  scanf("%d",&a[i][j]);  }  }  x=(a[1][1]\*a[2][2])-(a[2][1]\*a[1][2]);  y=(a[1][0]\*a[2][2])-(a[2][0]\*a[1][2]);  z=(a[1][0]\*a[2][1])-(a[2][0]\*a[1][1]);  det=(a[0][0]\*x)-(a[0][1]\*y)+(a[0][2]\*z);  printf("Determine=%d",det) ;  return 0;  } | \* |
|  | WAP that will take inputs of a n sized square matrix into a 2D array. Now show all the elements of its two diagonals. Reference: <http://en.wikipedia.org/wiki/Main_diagonal>   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 5  1 2 3 4 5  5 4 3 2 1  2 2 2 2 2  6 7 8 9 0  1 9 3 7 4 | Major diagonal: 1 4 2 9 4  Minor diagonal: 5 2 2 7 1 |   #include <stdio.h>  int main(){  int n,i,j;  scanf("%d",&n);  int a[n][n];  printf("Enter the array of the matrix:\n ");  for(i=0;i<n;i++){  for(j=0;j<n;j++){  scanf("%d",&a[i][j]) ;  }  }  printf("Major Diagonal:\n");  for(i=0;i<n;i++)  {  for(j=0;j<n;j++)  {  if(i==j){  printf("%d ",a[i][j]);  }  }  }  printf("\nMinor Diagonal:\n");  for(i=0;i<n;i++)  {  for(j=0;j<n;j++)  {  if(i+j==n-1)  {  printf("%d ",a[i][j]);  }  }  }  return 0;  } | \* |
|  | WAP that will take the size of an identity matrix from the user and generate the identity matrix into a 2D array. Finally display it. Reference:<http://en.wikipedia.org/wiki/Identity_matrix>   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 5 | 1 0 0 0 0  0 1 0 0 0  0 0 1 0 0  0 0 0 1 0  0 0 0 0 1 |   #include <stdio.h>  int main(){  int n,i,j;  scanf("%d",&n);  int a[n][n];  for(i=0;i<n;i++){  for(j=0;j<n;j++){  if(i==j){  a[i][j]=1;  }  else{  a[i][j]=0;  }  printf("%d ",a[i][j]) ;  }  printf("\n");  }  return 0;  } | \* |
|  | WAP that will take inputs of two *m x n* sized matrix into two 2D array, suppose A and B. Now do C = A + B. Finally display all the elements from matrix / 2D array C.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 2 3  1 2 3  2 3 4  1 1 1  2 2 2 | 2 3 4  4 5 6 |   #include <stdio.h>  int main()  {  int i,j,row,col;  int a[10][10],b[10][10],c[10][10];  printf("Enter the number of row n col: ");  scanf("%d %d",&row,&col);  printf("Enter the value of A matrix:\n ");  for(i=0; i<row; i++)  {  for(j=0; j<col; j++)  {  scanf("%d",&a[i][j]);  printf("%d ",a[i][j]);  }  printf("\n");  }  printf("Enter the value of B matrix:\n ");  for(i=0; i<row; i++)  {  for(j=0; j<col; j++)  {  scanf("%d",&b[i][j]);  printf("%d ",b[i][j]);  }  printf("\n");  }  printf("\n");  for(i=0; i<row; i++)  {  for(j=0; j<col; j++)  {  c[i][j]=a[i][j] + b[i][j];  printf("%d ",c[i][j]);  }  printf("\n");  }  return 0;  } | \* |
|  | WAP that will take inputs of two *3 x 3* sized matrix into two 2D array, suppose A and B. Now do C = A \* B (multiplication). Finally display all the elements from matrix / 2D array C.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 1 2 3  4 5 6  7 8 9  2 2 2  2 2 2  1 1 1 | 9 9 9  24 24 24  39 39 39 |   #include <stdio.h>  int main()  {  int i,j,k,sum=0;  int a[3][3],b[3][3],c[3][3];  printf("Enter the value of A matrix:\n ");  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  scanf("%d",&a[i][j]);  printf("%d ",a[i][j]);  }  printf("\n");  }  printf("Enter the value of B matrix:\n ");  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  scanf("%d",&b[i][j]);  printf("%d ",b[i][j]);  }  printf("\n");  }  printf("\n");  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  for(k=0;k<3;k++){  printf("%d ",c[i][j]);  sum=sum + a[i][k]\*b[k][j];  }  c[i][j]=sum;  z sum=0;  }  }  printf("Result matrix:\n ");  for(i=0; i<3; i++)  {  for(j=0; j<3; j++)  {  printf("%d ",c[i][j]);  }  printf("\n");  }  return 0;  } | \*\*\* |
|  | WAP that will take inputs of *m x n* sized matrix into a 2D array and find the maximum element with index locationfrom that matrix.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 3 3  1 2 3  4 5 6  2 9 2 | Max: 9  Location: [2][1] | | 2 3  9 8 7  3 4 5 | Max: 9  Location: [0][0] |   #include <stdio.h>  int main(){  int row,col,i,j,m,n,max;  scanf("%d %d",&row,&col);  int a[row][col];  printf("enter the values of the matrix:\n ");  for(i=0;i<row;i++){  for(j=0;j<col;j++){  scanf("%d",&a[i][j]);  printf("%d ",a[i][j]);  }  printf("\n");  }  max=a[0][0];  for(i=0;i<row;i++)  {  for(j=0;j<col;j++)  {  if(a[i][j]>max){  max=a[i][j];  m=i;  n=j;  }  }  }  printf("Max: %d\n",max);  printf("Location:[%d][%d]",m,n);  return 0;  } | \* |
|  | WAP that will take (n x n) integer inputs into a square matrix of dimension n (where n must be an odd number). Then calculate sum of the integers at first row, last row and two diagonals without overlap. Please see the sample input-output.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 5  1 2 3 4 5  2 3 4 1 6  3 4 9 6 7  4 2 6 7 8  5 4 3 2 1 | 52 | | 7  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1 | 23 |   #include <stdio.h>  int main(){  int n,i,j;  int sum1=0,sum2=0,sum3=0,sum;  printf("Enter odd number: ");  scanf("%d",&n);  int a[20][20];  printf("Enter the array of the matrix:\n ");  for(i=0;i<n;i++){  for(j=0;j<n;j++){  scanf("%d",&a[i][j]) ;  }  }  for(i=0;i<n;i++){  for(j=0;j<n;j++){  if(i==0 || i==(n-1)){  sum1=sum1+a[i][j];  }  }  }  for(i=1;i<=n-2;i++){  sum2=sum2+a[i][i];  sum3=sum3+a[i][n-1-i];  if(i==(n/2)){  sum3=sum3-a[i][i];  }  }  sum=sum1+sum2+sum3;  printf("sum=%d",sum);  return 0;  } | \*\* |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | WAP that will take (n x n) integer inputs into a square matrix of dimension n (where n must be an odd number). Then calculate sum of the integers based on following position pattern (consider only the boxed position during the sum). Please see the input-output.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 5  1 2 3 4 5  2 3 4 1 6  3 4 9 6 7  4 2 6 7 8  5 4 3 2 1 | 71 | | 7  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1 | 25 | | \*\* |
|  | WAP that will take (n x n) integer inputs into a square matrix of dimension n (where n must be an odd number). Then calculate sum of the integers based on following position pattern (consider only the boxed position during the sum). Please see the input-output.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 5  1 2 3 4 5  2 3 4 1 6  3 4 9 6 7  4 2 6 7 8  5 4 3 2 1 | 65 | | 7  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1  1 1 1 1 1 1 1 | 33 |   #include <stdio.h>  int main(){  int n,i,j;  int sum1=0,sum2=0,sum;  printf("Enter odd number: ");  scanf("%d",&n);  int a[20][20];  printf("Enter the array of the matrix:\n ");  for(i=0;i<n;i++){  for(j=0;j<n;j++){  scanf("%d",&a[i][j]) ;  }  }  printf("\n");  for(i=0;i<n;i++){  for(j=0;j<n;j++){  if(i%2!=0){  sum1=sum1+a[i][j];  }  }  }  for(i=0;i<n;i++){  for(j=0;j<n;j++){  if(i%2==0 && j%2!=0){  sum2=sum2+a[i][j];  }  }  }  sum=sum1+sum2;  printf("sum=%d",sum);  return 0;  } | \*\* |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | WAP that will take (m x n) integer inputs into a matrix of dimension m x n. Now reverse that matrix within itself and display it. Reversal means swap 1st column with the nth column, swap 2nd column with the (n-1)th column and so on…   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 3 3  1 2 3  4 5 6  2 9 2 | 3 2 1  6 5 4  2 9 2 | | 2 6  1 2 3 4 5 6  9 8 7 6 5 4 | 6 5 4 3 2 1  4 5 6 7 8 9 | | \*\* |
|  | WAP that will take (n x n) integer inputs into a square matrix of dimension n. Now determine whether the matrix is symmetric or not.  Reference: <http://en.wikipedia.org/wiki/Symmetric_matrix>   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 3  1 7 3  7 4 5  3 5 6 | Yes | | 2  1 3  4 2 | No |   #include <stdio.h>  int main()  {  int n,i,j,flag=1;  scanf("%d",&n);  int a[n][n],b[n][n];  printf("Enter the array of the matrix:\n ");  for(i=0; i<n; i++)  {  for(j=0; j<n; j++)  {  scanf("%d",&a[i][j]);  }  }  for(i=0; i<n; i++)  {  for(j=0; j<n; j++)  {  b[j][i]=a[i][j];  flag=1;  }  }  for(i=0; i<n; i++)  {  for(j=0; j<n; j++)  {  if(a[i][j]!=b[i][j])  {  flag++;  break;  }  }  }  if (flag==1){  printf("Yes");  }  else{  printf("No");  }  return 0;  } | \*\* |
|  | WAP that will take (m x n) positive integer inputs into a matrix of dimension m x n. Now replace all the duplicate integers by -1 in that matrix. Finally display it.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 3 3  1 7 3  7 4 5  3 5 6 | 1 7 3  -1 4 5  -1 -1 6 | | 2 6  2 2 2 2 2 2  6 5 4 3 2 1 | 2 -1 -1 -1 -1 -1  6 5 4 3 -1 1 |   #include <stdio.h>  int main()  {  int row,col,i,j,k,count=0;  int flag = 0;  scanf("%d %d",&row,&col);  int a[row][col],b[30];  printf("Enter the array of the matrix:\n ");  for(i=0; i<row; i++)  {  for(j=0; j<col; j++)  {  scanf("%d",&a[i][j]) ;  }  }  for(i=0; i<row; i++)  {  for(j=0; j<col; j++)  {  for(k=0; k<count+1; k++)  {  if(b[k]==a[i][j])  {    a[i][j]=-1;  flag = 1;  break ;  }  else{  flag = 0;  }  }  if(flag == 0)  {  b[count]=a[i][j];  count++;  }  }  }  printf("\n");  for(i=0; i<row; i++)  {  for(j=0; j<col; j++)  {  printf(" %d ",a[i][j]) ;  }  printf("\n");  }  return 0;  } | \*\*\* |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | WAP that will take (m x n) integer inputs into a matrix of dimension m x n. Now just simply add all the integers in that matrix and show the result.   |  |  | | --- | --- | | **Sample input** | **Sample output** | | 3 3  1 7 3  7 4 5  3 5 6 | 41 | | 2 6  2 2 2 2 2 2  6 5 4 3 2 1 | 33 |   #include <stdio.h>  int main(){  int row,col,i,j,sum=0;  scanf("%d %d",&row,&col);  int a[row][col];  printf("Enter the array of the matrix:\n ");  for(i=0;i<row;i++){  for(j=0;j<col;j++){  scanf("%d",&a[i][j]) ;  }  }  for(i=0;i<row;i++){  for(j=0;j<col;j++){  sum=sum+a[i][j];  }  }  printf("%d",sum);  return 0;  } | \* |