*HARSH KASHYAP  
CSE 4*

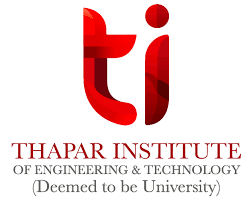
*101917088*

[*hkashyap\_be19@thapar.edu*](mailto:hkashyap_be19@thapar.edu)

A Practical activity Report submitted

for Data Structures (UCS301)

**DATA STRUCTURES**

****

Computer Science and Engineering

Patiala Campus

**2020**

Submitted to

Maninder Kaur

**Assignment 1A**

**Question 1 a)**

**Space required to store any two-dimensional array is a number of row X number of columns, Assuming array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.**

**(a) Diagonal Matrix.**

**SOLUTION CODE**

**/\* #Space required to store any two-dimensional array is**

**#. Assuming array is used to store elements of the following**

**#matrices, implement an efficient way that reduces the space requirement.**

**#(a) Diagonal Matrix**

**\*/**

**import java.util.Scanner;**

**class ass1a\_q1**

**{**

**static Scanner scr = new Scanner(System.in);**

**public static void main(final String[] args) {**

**System.out.println("Enter rows and columns for matrix (nXn) : ") ;**

**int diag = scr.nextInt();**

**int d[] = new int[diag];**

**int i,j;**

**for (i = 0; i < diag; i++) {**

**System.out.println("Enter value for diagonal ["+i+","+i+"] th element : ");**

**d[i] = scr.nextInt();**

**}**

**System.out.println("\nDiagonal Matrix");**

**for (i=0;i<diag;i++)**

**{**

**for (j=0;j<diag;j++)**

**{**

**if (i==j)**

**System.out.print(d[i]+" ");**

**else**

**System.out.print("0 ");**

**}**

**System.out.println();**

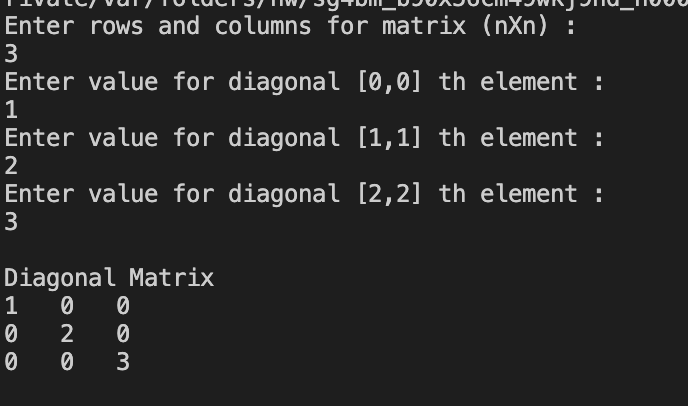
**}**

**System.out.println();**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 1 b)**

**Space required to store any two-dimensional array is a number of row X number of column, Assuming array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.**

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

**SOLUTION CODE**

**/\* #Space required to store any two-dimensional array is**

**#. Assuming array is used to store elements of the following**

**#matrices, implement an efficient way that reduces the space requirement.**

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

**\*/**

**import java.util.Scanner;**

**class ass1a\_q1b**

**{**

**static Scanner scr = new Scanner(System.in);**

**public static void main(final String[] args) {**

**System.out.println("Enter rows and columns for matrix (nXn) : ") ;**

**int diag = scr.nextInt();**

**int ud[]= new int[diag-1];;**

**int d[]=new int[diag];**

**int ld[]= new int[diag-1];**

**int i,j;**

**System.out.println("\nUPPER DIAGONAL");**

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

**System.out.println("Enter value for diagonal ["+i+","+(i+1)+"] th element : ");**

**ud[i] = scr.nextInt();**

**}**

**System.out.println("\nDIAGONAL");**

**for (i = 0; i < diag; i++) {**

**System.out.println("Enter value for diagonal ["+i+","+i+"] th element : ");**

**d[i] = scr.nextInt();**

**}**

**System.out.println("\nLOWER DIAGONAL");**

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

**System.out.println("Enter value for diagonal ["+(i+1)+","+i+"] th element : ");**

**ld[i] = scr.nextInt();**

**}**

**int u=0;**

**int l=0;**

**System.out.println("\nTRI DIAGONAL MATRIX");**

**for (i = 0; i < diag; i++) {**

**for (j = 0; j< diag; j++) {**

**if (i-j==-1)**

**System.out.print(ud[u++]+" ");**

**else if(i==j)**

**System.out.print(d[i]+" ");**

**else if (i-j==1)**

**System.out.print(ld[l++]+" ");**

**else**

**System.out.print("0"+" ");**

**}**

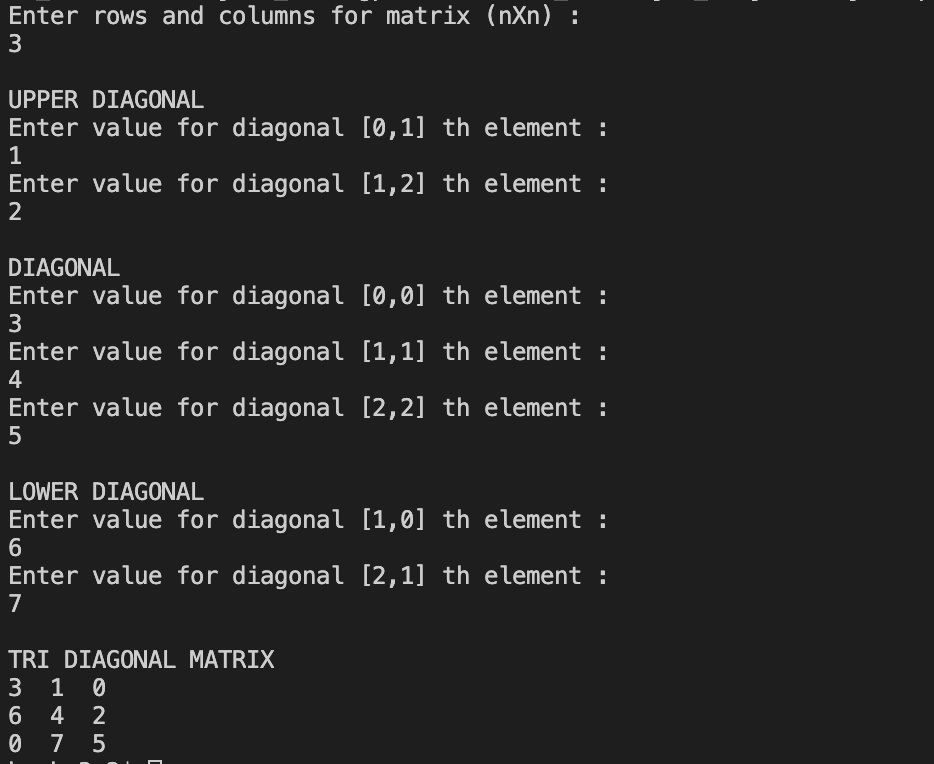
**System.out.println();**

**}**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 1 c)**

**Space required to store any two-dimensional array is a number of row X number of column, Assuming array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.**

**(c) Lower triangular Matrix.**

**SOLUTION CODE**

**/\* #Space required to store any two-dimensional array is**

**#. Assuming array is used to store elements of the following**

**#matrices, implement an efficient way that reduces the space requirement.**

**#(b) Lower triangular Matrix.**

**\*/**

**import java.util.Scanner;**

**class ass1a\_q1d**

**{**

**static Scanner scr = new Scanner(System.in);**

**public static void main(final String[] args) {**

**System.out.println("Enter rows and columns for matrix (nXn) : ") ;**

**int n = scr.nextInt();**

**int p = (n\*(n-1)/2);**

**int d[]=new int[p];**

**int i,j;**

**System.out.println("\nLOWER TRIANGULAR MATRIX");**

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

**System.out.println("Enter element : ");**

**d[i] = scr.nextInt();**

**}**

**int u =0;**

**System.out.println("\nMATRIX");**

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

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

**if (j<i)**

**System.out.print(d[u++]+" ");**

**else**

**System.out.print("0"+" ");**

**}**

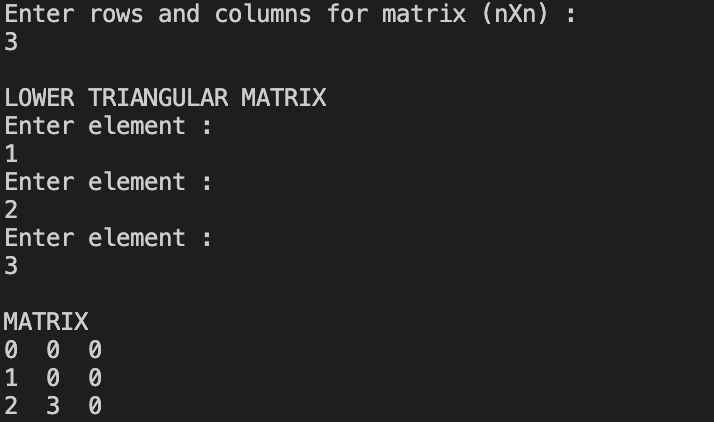
**System.out.println();**

**}**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 1 d)**

**Space required to store any two-dimensional array is a number of row X number of column, Assuming array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.**

**(c) Upper triangular Matrix.**

**SOLUTION CODE**

**//#(c) Upper triangular Matrix.**

**import java.util.Scanner;**

**class ass1a\_q1c**

**{**

**static Scanner scr = new Scanner(System.in);**

**public static void main(final String[] args) {**

**System.out.println("Enter rows and columns for matrix (nXn) : ") ;**

**int n = scr.nextInt();**

**int p = (n\*(n-1)/2);**

**int d[]=new int[p];**

**int i,j;**

**System.out.println("\nUPPER TRIANGULAR MATRIX");**

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

**System.out.println("Enter element : ");**

**d[i] = scr.nextInt();**

**}**

**int u =0;**

**System.out.println("\nMATRIX");**

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

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

**if (j>i)**

**System.out.print(d[u++]+" ");**

**else**

**System.out.print("0"+" ");**

**}**

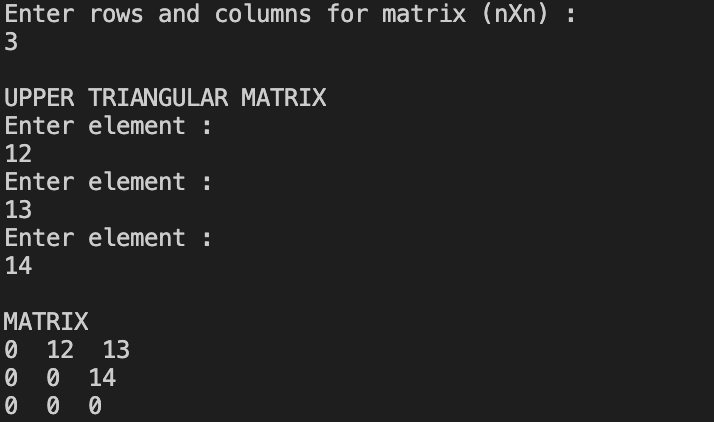
**System.out.println();**

**}**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 1 e)**

**Space required to store any two-dimensional array is a number of row X number of column, Assuming array is used to store elements of the following matrices, implement an efficient way that reduces the space requirement.**

**(e) Symmetric Matrix**

**SOLUTION CODE**

**import java.util.\*;**

**import java.io.\*;**

**class a1ques1e**

**{**

**static Scanner scr = new Scanner(System.in);**

**static int a[][];**

**public static void main(final String[] args) {**

**System.out.println("Enter rows and columns for matrix (nXn) : ") ;**

**int n = scr.nextInt();**

**int p = (n\*(n-1)/2);**

**int d[]=new int[n];**

**int ud[]= new int[p];**

**int dn=0;**

**int udn=0;**

**int ldn=0;**

**int i,j;**

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

**{**

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

**{**

**if(i==j)**

**{**

**System.out.println("Enter value of [" + i + ","+j+"] th element ");**

**d[dn++] = scr.nextInt();**

**}**

**if (j>i)**

**{**

**System.out.println("Enter value of [" + i + ","+j+"] th element ");**

**ud[udn++] = scr.nextInt();**

**}**

**}**

**}**

**dn=udn=ldn=0;**

**boolean ans;**

**System.out.println("\nSYMMETRIC MATRIX ");**

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

**{**

**ans= false;**

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

**{**

**if(i==j)**

**System.out.print(d[dn++]+" ");**

**else if (j>i)**

**System.out.print(ud[udn++]+" ");**

**else if (i>j && !ans)**

**for (int k=0,z=i-1,c=n-2;k<i;k++,z+=c--)**

**{**

**ans= true;**

**System.out.print(ud[z]+" ");**

**}**

**}**

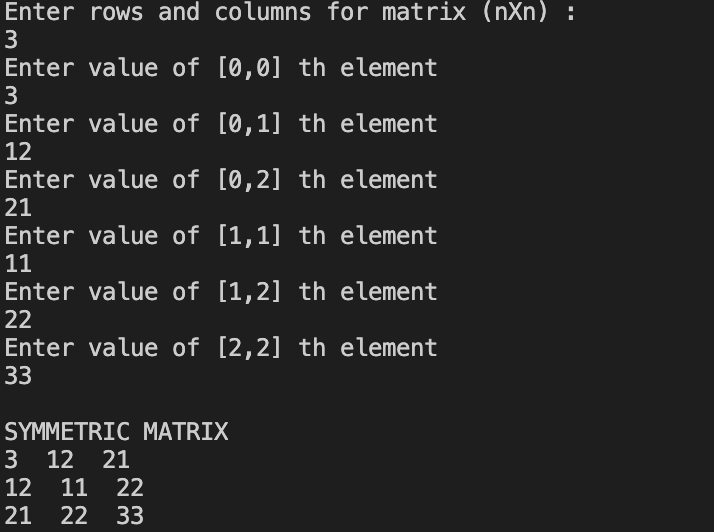
**System.out.println("");**

**}**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 2**

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

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

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

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

**SOLUTION CODE**

**/\*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.**

**\*/**

**import java.util.\*;**

**public class sparse\_matrix {**

**static int len=0;**

**static int counter=0;**

**static Scanner scr =new Scanner (System.in);**

**// insert elements into sparse matrix**

**private static void insert(int A[][],int r, int c,int n)**

**{**

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

**{**

**System.out.println("Enter value of "+(i)+"th non zero element");**

**A[i][2] = scr.nextInt();**

**System.out.println("Enter the row");**

**A[i][0]= scr.nextInt();**

**System.out.println("Enter the column");**

**A[i][1]= scr.nextInt();**

**}**

**A[n][0]=A[n][1]=A[n][2]=-1;**

**}**

**private static int[][] transpose(int A[][] ,int n)**

**{**

**len=0;**

**int trans[][] = new int[n+1][3];**

**trans[n][0]=trans[n][1]=trans[n][2]=-1;**

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

**trans[len][0]= A[len][1];**

**trans[len][1]= A[len][0];**

**trans[len][2]= A[len][2];**

**}**

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

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

**if (trans[j][0] > trans[j+1][0])**

**{**

**int temp = trans[j][0];**

**trans[j][0] = trans[j+1][0];**

**trans[j+1][0] = temp;**

**temp = trans[j][1];**

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

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

**temp = trans[j][2];**

**trans[j][2] = trans[j+1][2];**

**trans[j+1][2] = temp;**

**}**

**return trans;**

**}**

**private static void add(int A[][],int B[][],int r,int c)**

**{**

**int l1=0,l2=0;**

**counter=0;**

**int res[][] = new int[A.length+B.length][3];**

**for (int i=0;i<A.length;i++)**

**{**

**for (int j=0;j<A[0].length;j++)**

**{**

**if (i==A[l1][0] && j==A[l1][1] && i==B[l2][0] && j==B[l2][1] )**

**{**

**res[counter][2]=A[l1++][2]+B[l2++][2];**

**res[counter][0]=i;**

**res[counter++][1]=j;**

**}**

**else if (i==A[l1][0] && j==A[l1][1])**

**{**

**res[counter][2]=A[l1++][2];**

**res[counter][0]=i;**

**res[counter++][1]=j;**

**}**

**else if (i==B[l2][0] && j==B[l2][1] )**

**{**

**res[counter][2]=B[l2++][2];**

**res[counter][0]=i;**

**res[counter++][1]=j;**

**}**

**}**

**}**

**res[counter][0]=res[counter][1]=res[counter][2] =-1;**

**print(res,r,c);**

**}**

**private static void multiply(int A[][], int B[][],int r1, int c1, int r2,int c2,int n1, int n2)**

**{**

**B = transpose(B,n2);**

**int apos, bpos;**

**int pro[][] = new int[r1\*r2][3];**

**int x=0;**

**for (apos = 0; apos < A.length;)**

**{**

**int r = A[apos][0];**

**for (bpos = 0; bpos < B.length;) {**

**int c = B[bpos][0];**

**int tempa = apos;**

**int tempb = bpos;**

**int sum = 0;**

**while (tempa < A.length && A[tempa][0] == r && tempb < B.length && B[tempb][0] == c)**

**{**

**if (A[tempa][1] < B[tempb][1])**

**tempa++;**

**else if (A[tempa][1] > B[tempb][1])**

**tempb++;**

**else**

**sum += A[tempa++][2] \* B[tempb++][2];**

**}**

**if (sum != 0)**

**{**

**pro[x][0]=r;**

**pro[x][1]=c;**

**pro[x][2]=sum;**

**x++;**

**}**

**while (bpos < B.length && B[bpos][0] == c)**

**bpos++;**

**}**

**while (apos < A.length && A[apos][0] == r)**

**apos++;**

**}**

**pro[x][0]=pro[x][1]=pro[x][2]=-1;**

**print(pro,r1,c2);**

**}**

**// printing matrix**

**private static void print(int arr[][],int row, int col)**

**{**

**len=0;**

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

**{**

**for (int j=0;j<col;j++)**

**{**

**if (!(arr[len][0]==i && arr[len][1]==j))**

**System.out.print("0 ");**

**else**

**System.out.print(arr[len++][2]+" ");**

**}**

**System.out.println();**

**}**

**System.out.println();**

**}**

**public static void main(String args[])**

**{**

**System.out.println("\tMATRIX A");**

**System.out.println("Enter number of row for matrix A ");**

**int r1 = scr.nextInt();**

**System.out.println("Enter number of column for matrix A ");**

**int c1= scr.nextInt();**

**System.out.println("Enter number of non zero elements of matrix A");**

**int n1= scr.nextInt();**

**int A[][]= new int[n1+1][3];**

**System.out.println();**

**insert(A,r1,c1,n1);**

**System.out.println("\nThe original matrix A is : ");**

**print(A,r1, c1);**

**System.out.println("\nThe transpose matrix A is : ");**

**int trans[][]=transpose(A,n1);**

**print(trans,c1,r1);**

**System.out.println("\tMATRIX B");**

**System.out.println("Enter number of row for matrix B ");**

**int r2 = scr.nextInt();**

**System.out.println("Enter number of column for matrix B ");**

**int c2= scr.nextInt();**

**System.out.println("Enter number of non zero elements of matrix B");**

**int n2= scr.nextInt();**

**int B[][]= new int[n2+1][3];**

**System.out.println();**

**insert(B,r2,c2,n2);**

**System.out.println("\nThe original matrix B is : ");**

**print(B,r2, c2);**

**System.out.println("\nFinding the sum... ");**

**if(r1==r2 && c1==c2)**

**{**

**System.out.println("\nThe sum matrix of A+B is : ");**

**add(A,B,r1, c1);**

**}**

**else**

**System.out.println("\nMatrix can't be added ");**

**System.out.println("\nFinding the Product... ");**

**if(c1==r2)**

**{**

**System.out.println("\nThe product matrix of AXB is : ");**

**multiply(A,B,r1, c1,r2,c2,n1,n2);**

**}**

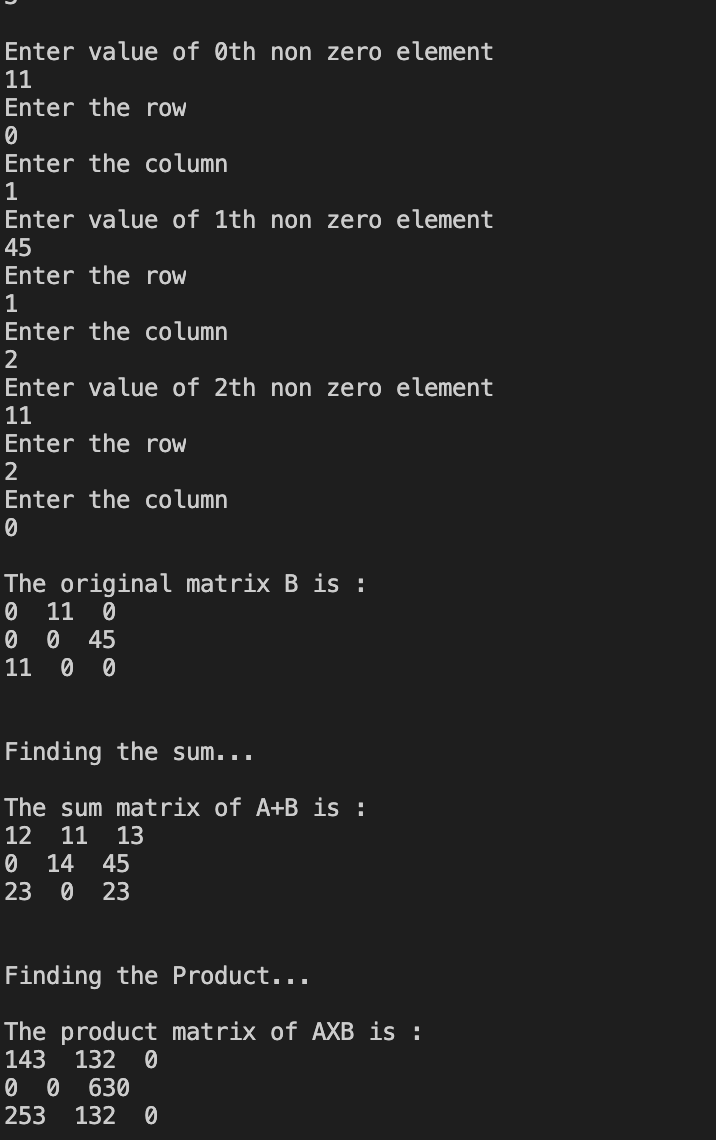
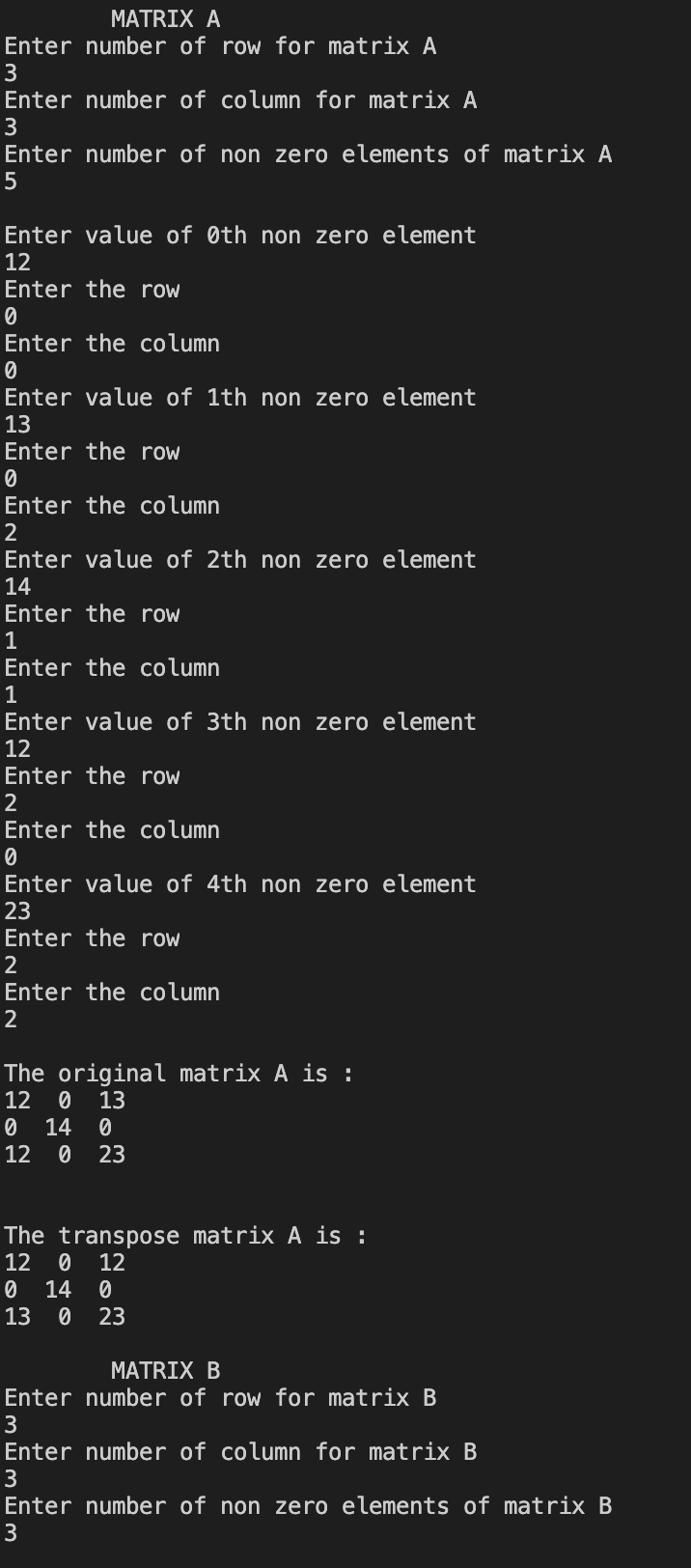
**else**

**System.out.println("\nMatrix can't be multiplied ");**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 3**

**Write a program to find the sum of every row and every column in a two-dimensional array.**

**SOLUTION CODE**

**//Write a program to find sum of every row and every column in a two-dimensional array**

**import java.io.\*;**

**import java.util.\*;**

**class SumofRowColumn**

**{**

**static Scanner scr = new Scanner(System.in);**

**public static void main(String[] args) {**

**int rows, cols, sumRow, sumCol;**

**System.out.println("Enter number of rows ");**

**int r = scr.nextInt();**

**System.out.println("Enter number of columns ");**

**int c = scr.nextInt();**

**int a[][] = new int[r][c];**

**for (int i = 0; i < a.length; i++) {**

**for (int j= 0; j < a[0].length; j++)**

**{**

**System.out.println("Enter value of [" + i + ","+j+"] th element ");**

**a[i][j] = scr.nextInt();**

**}**

**}**

**//Calculates number of rows and columns present in given matrix**

**rows = a.length;**

**cols = a[0].length;**

**//Calculates sum of each row of given matrix**

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

**sumRow = 0;**

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

**sumRow = sumRow + a[i][j];**

**}**

**System.out.println("Sum of " + (i+1) +" row: " + sumRow);**

**}**

**//Calculates sum of each column of given matrix**

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

**sumCol = 0;**

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

**sumCol = sumCol + a[j][i];**

**}**

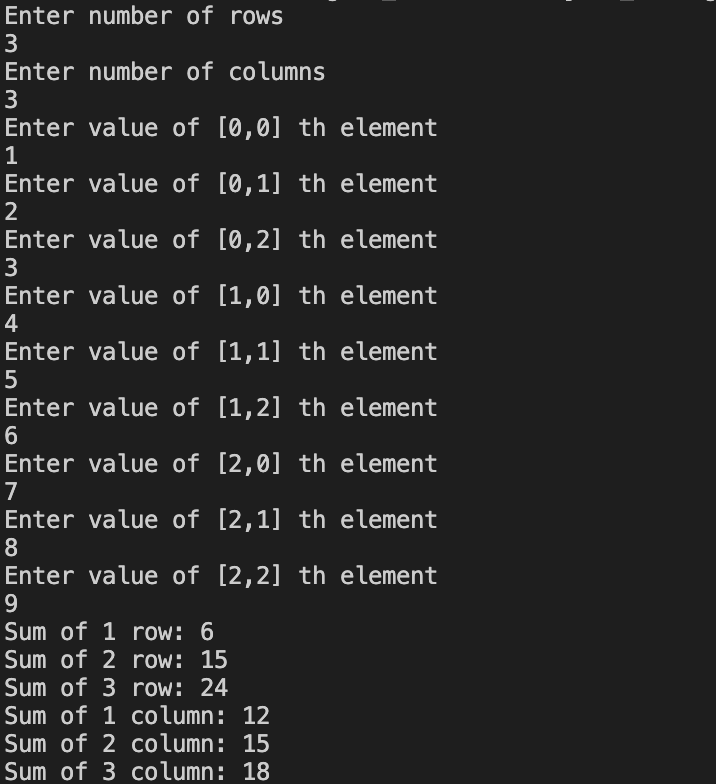
**System.out.println("Sum of " + (i+1) +" column: " + sumCol);**

**}**

**}**

**}**

***OUTPUT***

******

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 4**

**Write a program to find a saddle point in a two-dimensional array. A saddle point in a numerical array is a number that is larger than or equal to every number in its column, and smaller than or equal to every number in its row.**

**SOLUTION CODE**

**// Write a program to find a saddle point in a two-dimensional array.**

**//A saddle point in a numerical array is a number that is larger than or equal to every number in its column,**

**//and smaller than or equal to every number in its row.**

**import java.util.\*;**

**class Test**

**{**

**static boolean findSaddlePoint(int mat[][], int n)**

**{**

**// Process all rows one by one**

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

**{**

**int min\_row = mat[i][0], col\_ind = 0;**

**for (int j = 1; j < n; j++)**

**{**

**if (min\_row > mat[i][j])**

**{**

**min\_row = mat[i][j];**

**col\_ind = j;**

**}**

**}**

**int k;**

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

**if (min\_row < mat[k][col\_ind])**

**break;**

**if (k == n)**

**{**

**System.out.println("Value of Saddle Point " + min\_row);**

**return true;**

**}**

**}**

**return false;**

**}**

**static Scanner scr = new Scanner(System.in);**

**public static void main(final String[] args) {**

**System.out.println("Enter rows and columns for matrix (nXn) : ") ;**

**int n = scr.nextInt();**

**int a[][] = new int[n][n];**

**int i,j;**

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

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

**System.out.println("Enter value for ["+i+","+j+"] th element : ");**

**a[i][j] = scr.nextInt();**

**}**

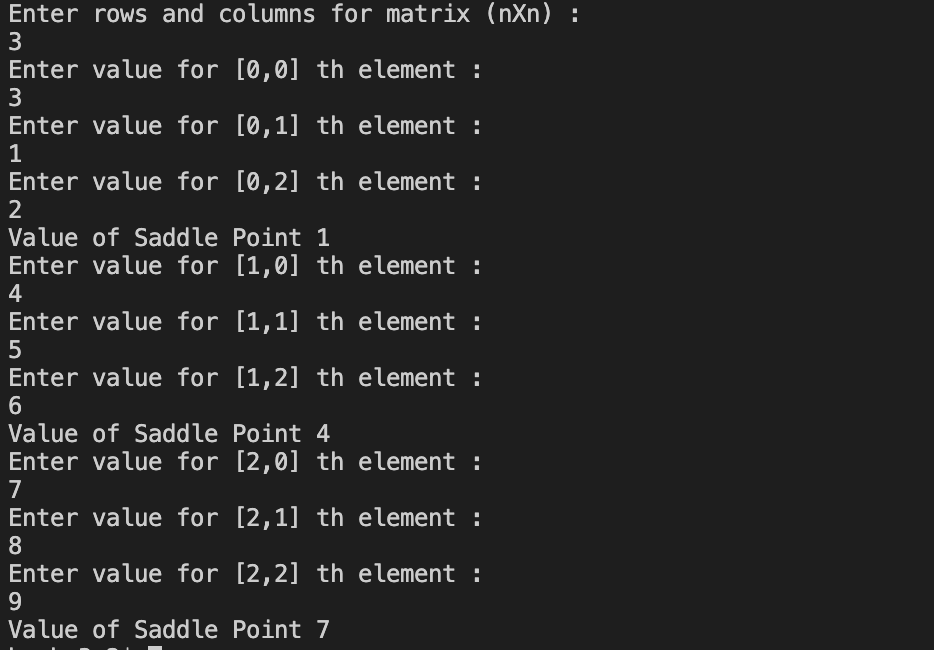
**if (findSaddlePoint(a, n) == false)**

**System.out.println("No Saddle Point ");**

**}**

**}**

**}**

**OUTPUT**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 5**

**Write a program to find a saddle point in a two-dimensional array. A saddle point in a numerical array is a number that is larger than or equal to every number in its column, and smaller than or equal to every number in its row.**

**SOLUTION CODE**

**// Java program to print a given matrix in spiral form**

**import java.io.\*;**

**import java.util.\*;**

**class a1ques5 {**

**static Scanner scr = new Scanner(System.in);**

**// Function print matrix in spiral form**

**static int[] spiralPrint(int m, int n, int a[][]) {**

**int b[] = new int [m\*n];**

**int x=0;**

**int i, k = 0, l = 0;**

**while (k < m && l < n) {**

**// Print the first row from the remaining rows**

**for (i = l; i < n; ++i) {**

**b[x++]=a[k][i];**

**}**

**k++;**

**// Print the last column from the remaining columns**

**for (i = k; i < m; ++i) {**

**b[x++]=a[i][n-1];**

**}**

**n--;**

**// Print the last row from the remaining rows \*/**

**if (k < m) {**

**for (i = n - 1; i >= l; --i) {**

**b[x++]=a[m - 1][i];**

**}**

**m--;**

**}**

**// Print the first column from the remaining columns \*/**

**if (l < n) {**

**for (i = m - 1; i >= k; --i) {**

**b[x++]=a[i][l];**

**}**

**l++;**

**}**

**System.out.println();**

**}**

**return b;**

**}**

**// driver program**

**public static void main(String[] args) {**

**System.out.println("Enter number of rows ");**

**int r = scr.nextInt();**

**System.out.println("Enter number of columns ");**

**int c = scr.nextInt();**

**int a[][] = new int[r][c];**

**for (int i = 0; i < a.length; i++) {**

**for (int j= 0; j < a[0].length; j++)**

**{**

**System.out.println("Enter value of [" + i + ","+j+"] th element ");**

**a[i][j] = scr.nextInt();**

**}**

**}**

**for (int i = 0; i < a.length; i++) {**

**for (int j= 0; j < a[0].length; j++)**

**{**

**System.out.print(a[i][j]+" ");**

**}**

**System.out.println();**

**}**

**int b[]=spiralPrint(r,c, a);**

**System.out.println("The matrix in spiral format is");**

**for (int i = 0; i < b.length; i++)**

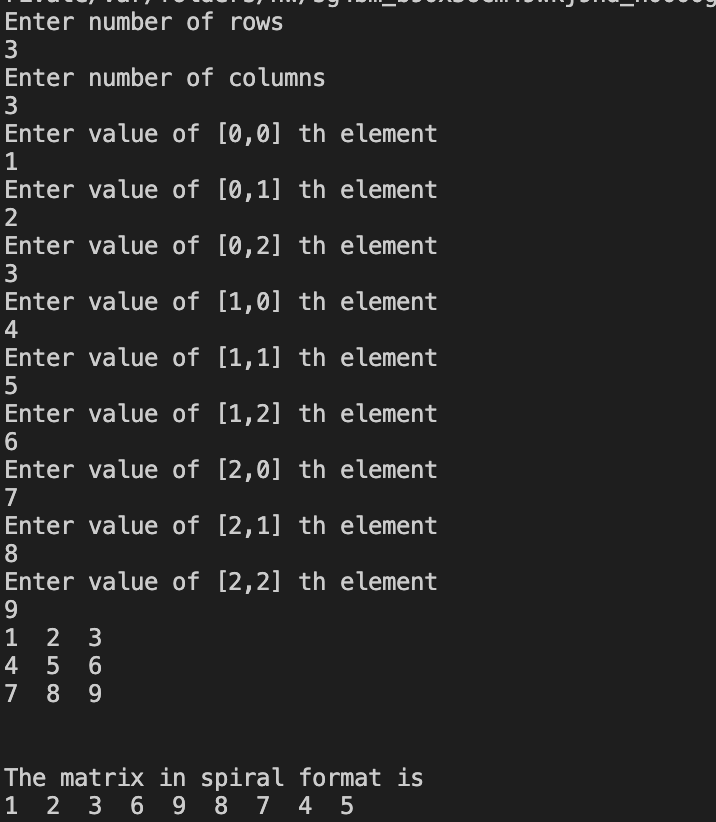
**System.out.print(b[i]+" ");**

**System.out.println();**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Question 5**

**Given a matrix of m \* n elements (m rows, n columns), return all elements of the matrix in spiral order.**

**SOLUTION CODE**

**// Java program Given an integer A, generate a square matrix filled with elements from 1 to A2 in spiral order.**

**import java.io.\*;**

**import java.util.\*;**

**class a1ques6 {**

**static Scanner scr = new Scanner(System.in);**

**static int a[][];**

**// Function print matrix in spiral form**

**static void spiral(int m, int n, int a[][]) {**

**System.out.println("\nThe matrix in spiral form :");**

**int i, k = 0, l = 0;**

**int z=1;**

**while (k < m && l < n) {**

**for (i = l; i < n; ++i) {**

**a[k][i]=z++;**

**}**

**k++;**

**for (i = k; i < m; ++i) {**

**a[i][n-1]=z++;**

**}**

**n--;**

**if (k < m) {**

**for (i = n - 1; i >= l; --i) {**

**a[m-1][i]=z++;**

**}**

**m--;**

**}**

**if (l < n) {**

**for (i = m - 1; i >= k; --i) {**

**a[i][l]=z++;**

**}**

**l++;**

**}**

**}**

**}**

**// driver program**

**public static void main(String[] args) {**

**System.out.println("Enter number for matrix (NXN) ");**

**int r = scr.nextInt();**

**a = new int[r][r];**

**spiral(r,r, a);**

**int c=0;**

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

**for (int j= 0; j < r; j++)**

**{**

**System.out.print(a[i][j]+" ");**

**}**

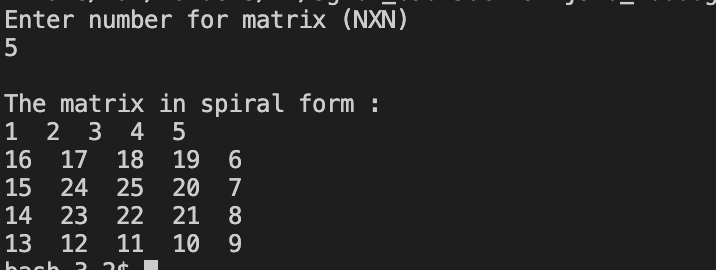
**System.out.println();**

**}**

**}**

**}**

**OUTPUT**

****

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**