ASSIGNMENT-1A

NAME – ANSHIKA

ROLL NO. – 102003183

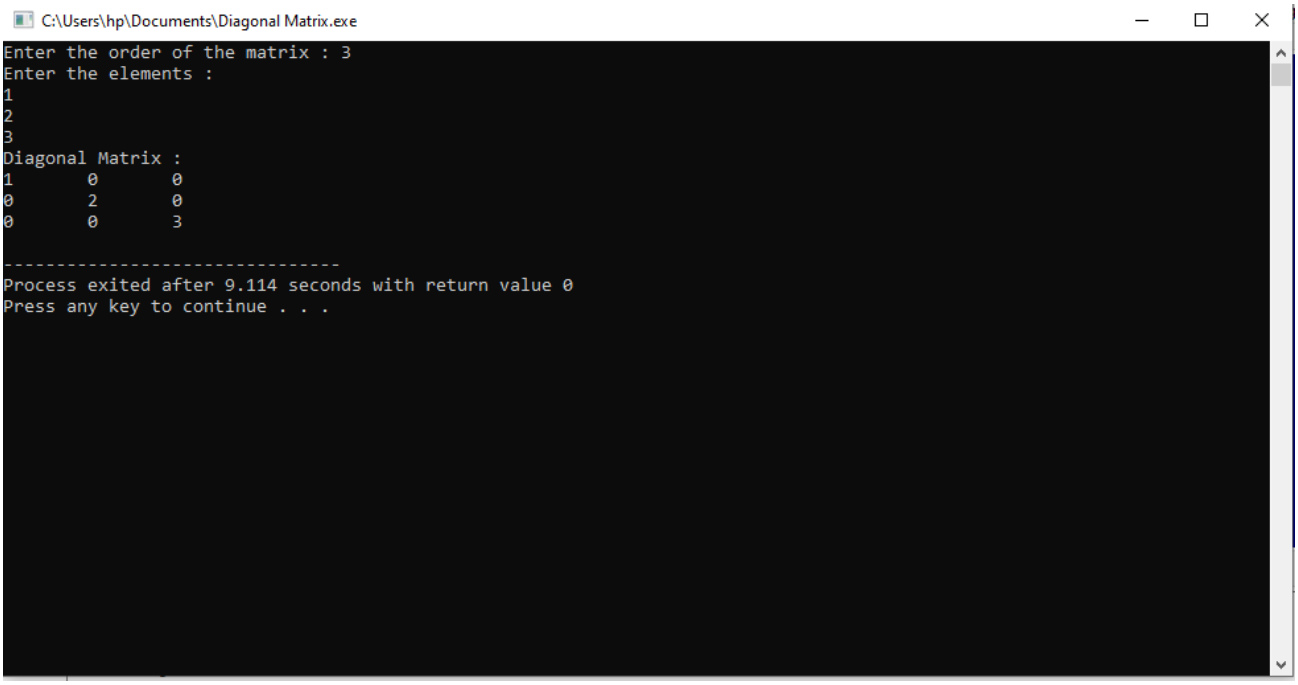
SUBGROUP – 2COE8

Question 1-

(a)
```
#include<iostream>
using namespace std;

int main()
{
	int n;
	cout<<"Enter the order of the matrix : ";
	cin>>n;
	int arr[n];
	cout<<"Enter the elements : "<<endl;
	for(int i=0; i<n; i++){
		cin>>arr[i];
	}
	cout<<"Diagonal Matrix : "<<endl;
	int k=0;
	for(int i=0; i<n; i++){
		for(int j=0; j<n; j++){
			if(i==j){
				cout<<arr[k++]<<"\t";
			}
			else{
				cout<<0<<"\t";
			}
		}
		cout<<endl;
```

```
        }
        return 0;
    }
```

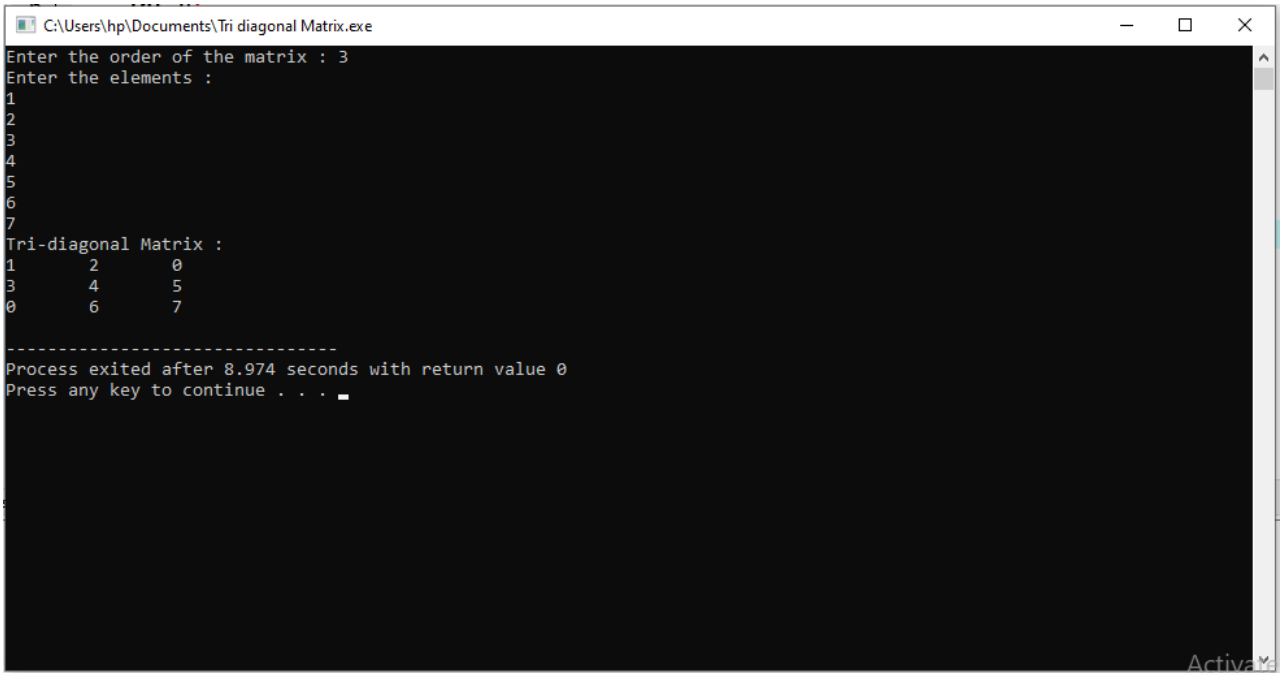

C:\Users\hp\Documents\Diagonal Matrix.exe
Enter the order of the matrix : 3
Enter the elements :
1
2
3
Diagonal Matrix :
1 0 0
0 2 0
0 0 3

Process exited after 9.114 seconds with return value 0
Press any key to continue . . .

(b)
```
#include<iostream>
using namespace std;

int main()
{
	int n;
	cout<<"Enter the order of the matrix : ";
	cin>>n;
	int size=n+2*(n-1);
	int arr[size];
	cout<<"Enter the elements : "<<endl;
	for(int i=0; i<size; i++){
		cin>>arr[i];
	}
	cout<<"Tri-diagonal Matrix : "<<endl;
	int k=0;
	for(int i=0; i<n; i++){
		for(int j=0; j<n; j++){
			if(i-j==-1||i-j==0||i-j==1){
				cout<<arr[k++]<<"\t";
			}
			else{
				cout<<0<<"\t";
			}
		}
		cout<<endl;
```

```
        }
        return 0;
    }
```

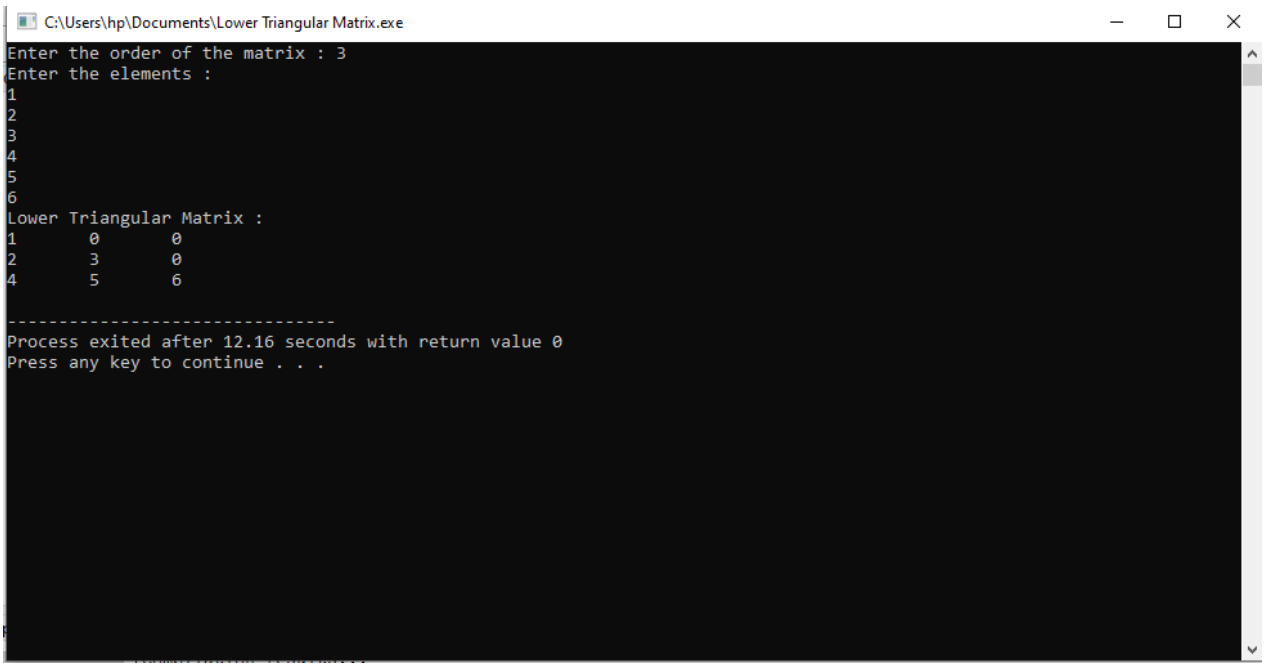

C:\Users\hp\Documents\Tri diagonal Matrix.exe
Enter the order of the matrix : 3
Enter the elements :
1
2
3
4
5
6
7
Tri-diagonal Matrix :
1 2 0
3 4 5
0 6 7

Process exited after 8.974 seconds with return value 0
Press any key to continue . . .

(c)
```
#include<iostream>
 using namespace std;


int main()
{
        int n;
        cout<<"Enter the order of the matrix : ";
        cin>>n;
        int size=n*(n+1)/2;
        int arr[size];
        cout<<"Enter the elements : "<<endl;
        for(int i=0; i<size; i++){
                cin>>arr[i];
        }
        cout<<"Lower Triangular Matrix : "<<endl;
        int k=0;
        for(int i=0; i<n; i++){
                for(int j=0; j<n; j++){
                        if(i>=j){
                                cout<<arr[k++]<<"\t";
                        }
                        else{
                                cout<<0<<"\t";
                        }
                }
                cout<<endl;
```

```
        }
        return 0;
    }
```

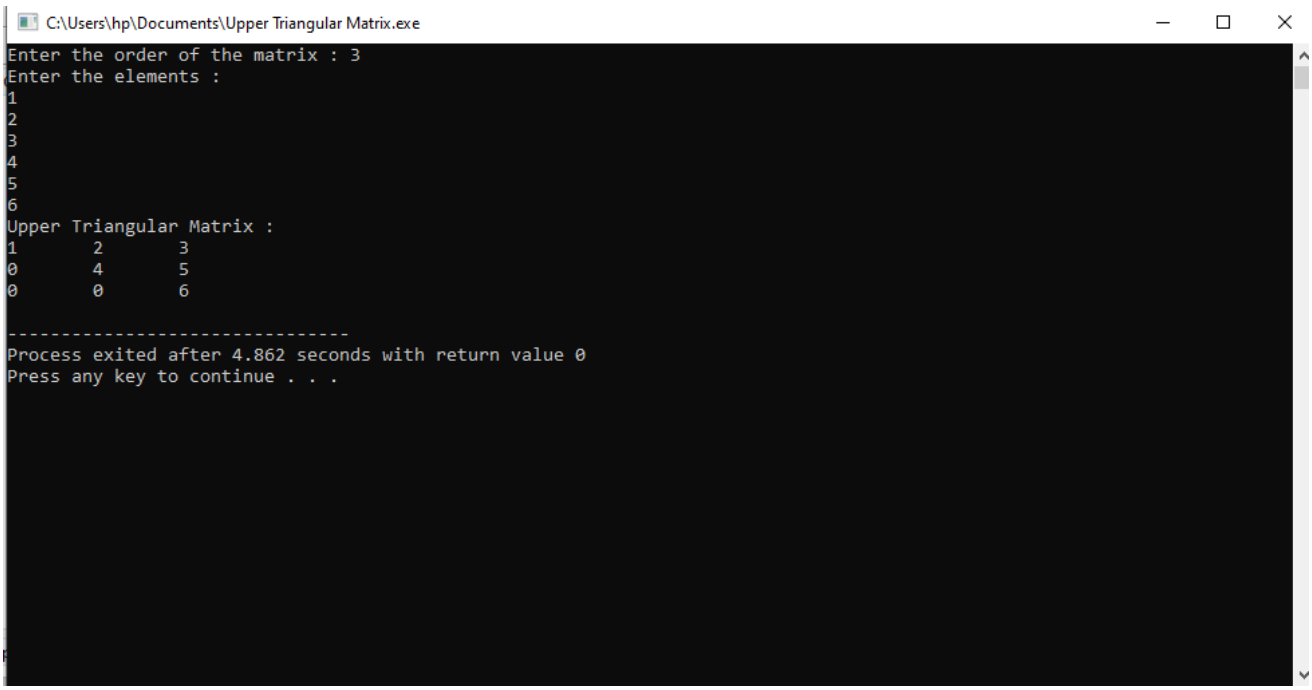

C:\Users\hp\Documents\Lower Triangular Matrix.exe
Enter the order of the matrix : 3
Enter the elements :
1
2
3
4
5
6
Lower Triangular Matrix :
1 0 0
2 3 0
4 5 6

Process exited after 12.16 seconds with return value 0
Press any key to continue . . .

(d)
```
#include<iostream>
using namespace std;

int main()
{
	int n;
	cout<<"Enter the order of the matrix : ";
	cin>>n;
	int size=n*(n+1)/2;
	int arr[size];
	cout<<"Enter the elements : "<<endl;
	for(int i=0; i<size; i++){
		cin>>arr[i];
	}
	cout<<"Upper Triangular Matrix : "<<endl;
	int k=0;
	for(int i=0; i<n; i++){
		for(int j=0; j<n; j++){
			if(i<=j){
				cout<<arr[k++]<<"\t";
			}
			else{
				cout<<0<<"\t";
			}
		}
		cout<<endl;
```

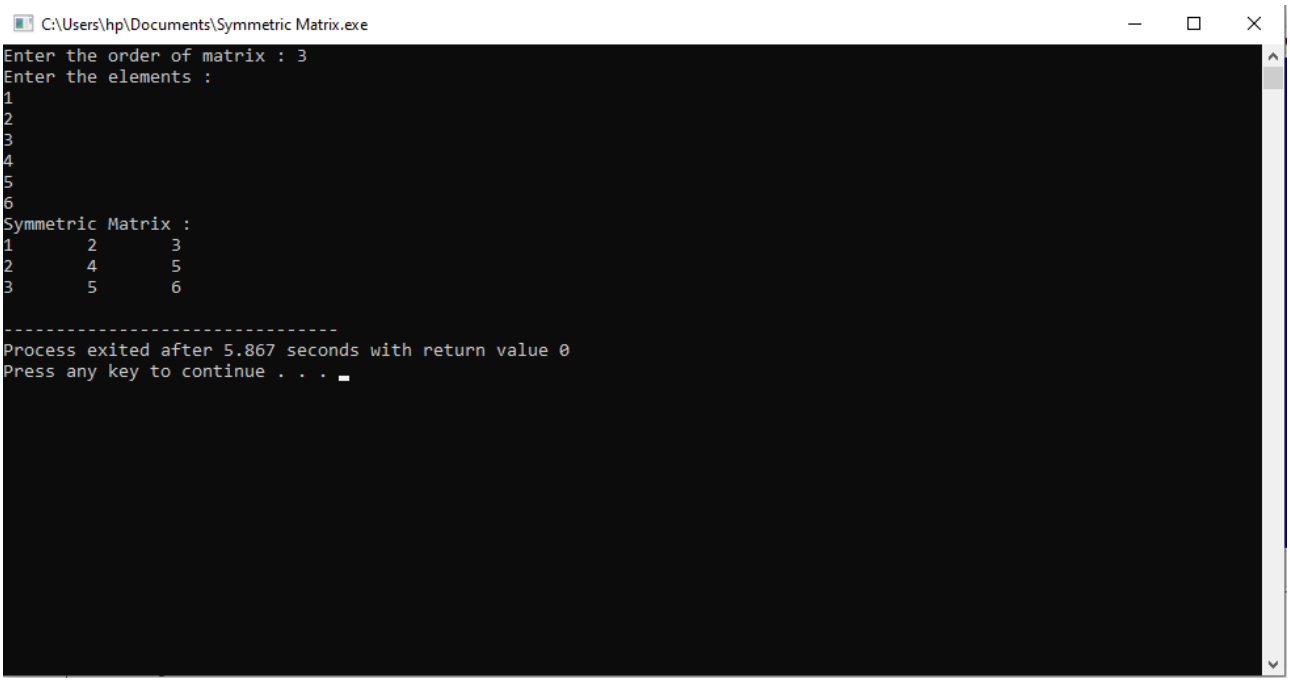
```
        }
        return 0;
    }
```


C:\Users\hp\Documents\Upper Triangular Matrix.exe
Enter the order of the matrix : 3
Enter the elements :
1
2
3
4
5
6
Upper Triangular Matrix :
1 2 3
0 4 5
0 0 6

Process exited after 4.862 seconds with return value 0
Press any key to continue . . .

(e)
```
#include<iostream>
using namespace std;
int main()
{
        int n,size,k=0,l=1;
        cout<<"Enter the order of matrix : ";
        cin>>n;
        size=n*(n+1)/2;
        int arr[size];
        cout<<"Enter the elements : "<<endl;
        for (int i=0; i<size; i++)
        {
          cin>>arr[i];
        }
        cout<<"Symmetric Matrix : "<<endl;
        for (int i=0; i<n; i++)
        {
          int diff=n-1,count=i;
          k=i;
          for (int j=0; j<n; j++)
          {
              cout<<arr[k]<<"\t";
              if(count>0)
              {
                k=k+diff;
                diff--;
```

```
                count--;
            }
            else
                k++;
        }
        cout<<endl;
    }
}
```



Question 2-

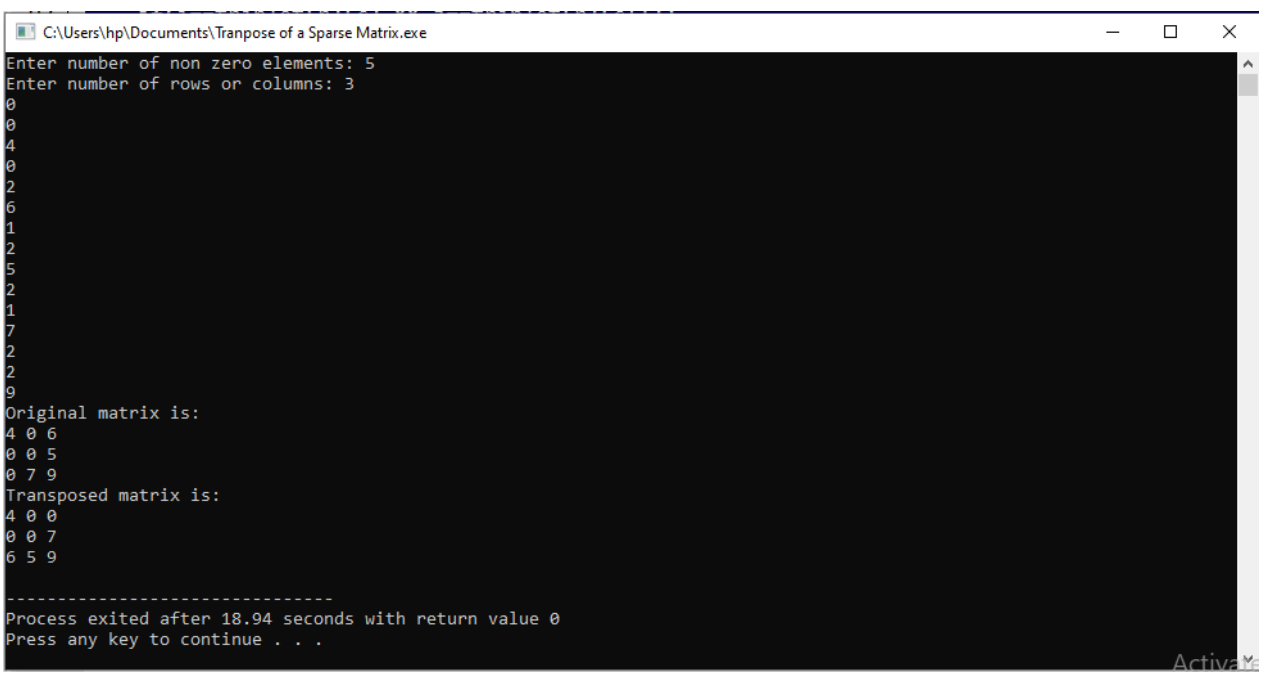
(a)
```
#include<iostream>
using namespace std;
int main()
{
	int n,i,a,j,r=1,c=0;
	cout<<"Enter number of non zero elements: ";
	cin>>n;
	cout<<"Enter number of rows or columns: ";
	cin>>a;
	int triplet[n+1][3];
	triplet[0][0]=a;
	triplet[0][1]=a;
	triplet[0][2]=n;
	for(i=1;i<n+1;i++)
	{
	for(j=0;j<3;j++)
	{
	cin>>triplet[i][j];
	}
	}
	int matrix[a][a];
	for(i=0;i<a;i++)
	{
	for(j=0;j<a;j++)
	{
```

```
if(i==triplet[r][c] && j==triplet[r][c+1])
{
matrix[i][j]=triplet[r][c+2];
r++;
}
else
{
matrix[i][j]=0;
}
}
}
cout<<"Original matrix is: "<<endl;
for(i=0;i<a;i++)
{
for(j=0;j<a;j++)
{
cout<<matrix[i][j];
cout<<" ";
}
cout<<endl;
}
// triplet in column major
int swap;
for(i=1;i<n+1;i++)
{
swap=triplet[i][0];
```

```
triplet[i][0]=triplet[i][1];
triplet[i][1]=swap;
}
// Sorting the given triplets
for(i=1;i<n+1;i++)
{
for(j=i+1;j<n+1;j++)
{
if(triplet[i][0]>triplet[j][0])
{
swap=triplet[i][0];
triplet[i][0]=triplet[j][0];
triplet[j][0]=swap;
swap=triplet[i][1];
triplet[i][1]=triplet[j][1];
triplet[j][1]=swap;
swap=triplet[i][2];
triplet[i][2]=triplet[j][2];
triplet[j][2]=swap;
}
}
}
for(i=1;i<n+1;i++)
{
for(j=i+1;j<n+1;j++)
{
```

```
if(triplet[i][1]>triplet[j][1] && triplet[i][0]==triplet[j][0])
{
swap=triplet[i][1];
triplet[i][1]=triplet[j][1];
triplet[j][1]=swap;
swap=triplet[i][2];
triplet[i][2]=triplet[j][2];
triplet[j][2]=swap;
}
}
}
cout<<"Transposed matrix is:"<<endl;
r=1;
c=0;
int transposed[a][a];
for(i=0;i<a;i++)
{
for(j=0;j<a;j++)
{
if(i==triplet[r][c] && j==triplet[r][c+1])
{
transposed[i][j]=triplet[r][c+2];
r++;
}
else
{
```

```
        transposed[i][j]=0;
        }
        }
        }
        for(i=0;i<a;i++)
        {
        for(j=0;j<a;j++)
        {
        cout<<transposed[i][j];
        cout<<" ";
        }
        cout<<endl;
        }
        return 0;
}
```



(b) #include<iostream>

```
using namespace std;
int main()
{
        int n,a,b,i,j,r=1,c=0,k;
        cout<<"Enter number of non zero elements in first matrix: ";
        cin>>a;
        cout<<"Enter number of non zero elements in second matrix: ";
        cin>>b;
        cout<<"Enter number of rows or columns: ";
        cin>>n;
        int firstt[a+1][3],secondt[b+1][3],firstm[n][n],secondm[n][n];
        firstt[0][0]=n;
        firstt[0][1]=n;
        firstt[0][2]=a;
        secondt[0][0]=n;
        secondt[0][1]=n;
        secondt[0][2]=b;
        cout<<"Enter triplet for first matrix:"<<endl;
        for(i=1;i<a+1;i++)
        {
        for(j=0;j<3;j++)
        {
        cin>>firstt[i][j];
        }
        }
```

```
cout<<"Enter triplet for second matrix:"<<endl;
for(i=1;i<b+1;i++)
{
for(j=0;j<3;j++)
{
cin>>secondt[i][j];
}
}
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(i==firstt[r][c] && j==firstt[r][c+1])
{
firstm[i][j]=firstt[r][c+2];
r++;
}
else
{
firstm[i][j]=0;
}
}
}
r=1,c=0;
for(i=0;i<n;i++)
{
```

```
for(j=0;j<n;j++)
{
if(i==secondt[r][c] && j==secondt[r][c+1])
{
secondm[i][j]=secondt[r][c+2];
r++;
}
else
{
secondm[i][j]=0;
}
}
}
cout<<"First matrix is:"<<endl;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
cout<<firstm[i][j];
cout<<" ";
}
cout<<endl;
}
cout<<"Second matrix is:"<<endl;
for(i=0;i<n;i++)
{
```

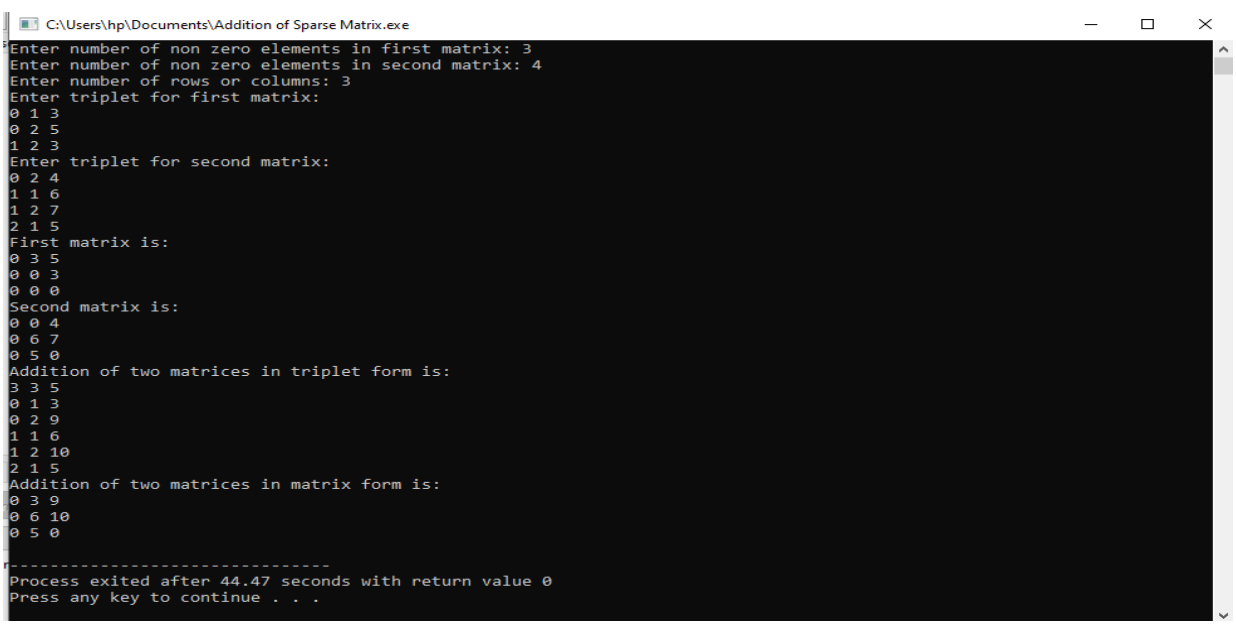
```
for(j=0;j<n;j++)
{
cout<<secondm[i][j];
cout<<" ";
}
cout<<endl;
}
int counter=0;
r=0;
c=1;
int add[20][3];
add[0][0]=n;
add[0][1]=n;
i=j=k=1;
while(i<=a && j<=b)
{
if(firstt[i][0]<secondt[j][0])
{
add[k][0]=firstt[i][0];
add[k][1]=firstt[i][1];
add[k][2]=firstt[i][2];
i++;
k++;
counter++;
}
else if(secondt[j][0]<firstt[i][0])
```

```
{
add[k][0]=secondt[j][0];
add[k][1]=secondt[j][1];
add[k][2]=secondt[j][2];
j++;
k++;
counter++;
}
else if(firstt[i][1]<secondt[j][1])
{
add[k][0]=firstt[i][0];
add[k][1]=firstt[i][1];
add[k][2]=firstt[i][2];
i++;
k++;
counter++;
}
else if(firstt[i][1]>secondt[j][1])
{
add[k][0]=secondt[j][0];
add[k][1]=secondt[j][1];
add[k][2]=secondt[j][2];
j++;
k++;
counter++;
}
```

```
else
{
add[k][0]=firstt[i][0];
add[k][1]=firstt[i][1];
add[k][2]=firstt[i][2]+secondt[j][2];
i++;
j++;
k++;
counter++;
}
}
while(i<=a)
{
add[k][0]=firstt[i][0];
add[k][1]=firstt[i][1];
add[k][2]=firstt[i][2];
i++;
k++;
counter++;
}
while(j<=b)
{
add[k][0]=secondt[j][0];
add[k][1]=secondt[j][1];
add[k][2]=secondt[j][2];
j++;
```

```
k++;
counter++;
}
add[0][2]=counter;
 cout<<"Addition of two matrices in triplet form is: "<<endl;
 for(i=0;i<=counter;i++)
 {
 for(j=0;j<3;j++)
 {
 cout<<add[i][j];
 cout<<" ";
}
cout<<endl;
}
int addm[n][n];
r=1;
for(i=0;i<n;i++)
{
for(j=0;j<n;j++)
{
if(i==add[r][0] && j==add[r][1])
{
addm[i][j]=add[r][2];
r++;
}
else
```

```
        {
        addm[i][j]=0;
        }
        }
        }
        cout<<"Addition of two matrices in matrix form is:"<<endl;
        for(i=0;i<n;i++)
        {
        for(j=0;j<n;j++)
        {
        cout<<addm[i][j]<<" ";
        }
        cout<<endl;
        }
        return 0;
}
```



(c)
```
#include<iostream>
#include<string>
using namespace std;
int main() {
	int r1,c1,n1;
	int r2,c2,n2;
	cout<<"Enter the dimensions of matrix 1: ";
	cin>>r1>>c1;
	cout<<"Enter the dimensions of matrix 2: ";
	cin>>r2>>c2;
	if(c1 != r2){
	cout<<"Can't multiply, Invalid dimensions\n";
	exit(0);
	}
	cout<<"\nMatrix 1:\n";
	cout<<"Enter the number of non-zero elements in the matrix: ";
	cin>>n1;
	int matrixA[n1][3];
	for(int i=0 ; i<n1 ; i++){
	cout<<"Enter the Row index , Column index and Value of element"<<i+1<<endl;
	cin>>matrixA[i][0];
	cin>>matrixA[i][1];
	cin>>matrixA[i][2];
	}
	cout<<"\nMatrix 2:\n";
	cout<<"Enter the number of non-zero elements in the matrix: ";
```

```
cin>>n2;
int matrixB[n2][3];
for(int i=0 ; i<n2 ; i++){
cout<<"Enter the Row index , Column index and Value of element"<<i+1<<endl;
cin>>matrixB[i][0];
cin>>matrixB[i][1];
cin>>matrixB[i][2];
}
cout<<endl;
cout<<"Matrix 1 :\n";
for(int i=0 ; i<n1 ; i++){
cout<<matrixA[i][0]<<" "<<matrixA[i][1]<<" "<<matrixA[i][2]<<endl;
}
cout<<"Matrix 2 :\n";
for(int i=0 ; i<n2 ; i++){
cout<<matrixB[i][0]<<" "<<matrixB[i][1]<<" "<<matrixB[i][2]<<endl;
}
// Transpose of matrix B:
int transposeB[n2][3];
int k=0;
for(int i=0 ; i<c2 ; i++){
for(int j=0 ; j<n2 ; j++){
if(matrixB[j][1] == i){
transposeB[k][0] = matrixB[j][1];
transposeB[k][1] = matrixB[j][0];
transposeB[k][2] = matrixB[j][2];
```

```
k++;
}
}
}
cout<<"\n\nTranspose of sparse matrix is :\n";
for(int i=0 ; i<n2 ; i++){
cout<<transposeB[i][0]<<" "<<transposeB[i][1]<<" "<<transposeB[i][2]<<endl;
}
// Multiplication of A and B:
int count=0;
int multiOfColumn = 1;
int sumOfCols = 0;
int multiNotZero;
int max = n1>n2 ? n1 : n2;
int matrixC[n1+n2][3];
for(int i=0 ; i<r1 ; i++){
for(int j=0 ; j<c2 ; j++){
sumOfCols = 0 ;
for(int c=0 ; c<c1 ; c++){
multiNotZero=0;
multiOfColumn = 1;
for(int k=0 ; k<max ; k++){
if(k<n1 && matrixA[k][0]==i && matrixA[k][1]==c){
multiOfColumn = multiOfColumn * matrixA[k][2];
multiNotZero++;
}
```

```
        if(k<n2 && matrixB[k][0]==j && matrixB[k][1]==c){
        multiOfColumn = multiOfColumn * matrixB[k][2];
        multiNotZero++;
        }
        }
        if(multiNotZero!=2){
        multiOfColumn = 0;
        }
        sumOfCols = sumOfCols + multiOfColumn;
        }
        if(sumOfCols != 0){
        matrixC[count][0] = i;
        matrixC[count][1] = j;
        matrixC[count][2] = sumOfCols;
        count++;
        }
        }
        }
        cout<<"AxB:\n\n";
        for(int i=0 ; i<count ; i++){
        cout<<matrixC[i][0]<<"\t"<<matrixC[i][1]<<"\t"<<matrixC[i][2]<<endl;
        }
        return 0;
}
```

C:\Users\hp\Documents\Multiplication of Sparse Matrix.exe

```
Enter the dimensions of matrix 1: 3
3
Enter the dimensions of matrix 2: 3
3

Matrix 1:
Enter the number of non-zero elements in the matrix: 4
Enter the Row index , Column index and Value of element1
0 1 3
Enter the Row index , Column index and Value of element2
0 2 5
Enter the Row index , Column index and Value of element3
1 1 6
Enter the Row index , Column index and Value of element4
2 1 7

Matrix 2:
Enter the number of non-zero elements in the matrix: 4
Enter the Row index , Column index and Value of element1
0 0 5
Enter the Row index , Column index and Value of element2
0 1 6
Enter the Row index , Column index and Value of element3
1 2 16
Enter the Row index , Column index and Value of element4
2 2 9
```

C:\Users\hp\Documents\Multiplication of Sparse Matrix.exe

```
Matrix 1 :
0 1 3
0 2 5
1 1 6
2 1 7
Matrix 2 :
0 0 5
0 1 6
1 2 16
2 2 9


Transpose of sparse matrix is :
0 0 5
1 0 6
2 1 16
2 2 9
AxB:

0       0       18
0       1       80
0       2       45
1       0       36
2       0       42

--------------------------------
Process exited after 48.98 seconds with return value 0
Press any key to continue . . .
```

Question 3-

```
#include<iostream>
using namespace std;

int main()
{
   int sumRow, sumCol;
   int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
   for(int i = 0; i<3; i++){
      sumRow = 0;
      for(int j = 0; j<3; j++){
        sumRow = sumRow + a[i][j];
      }
   }
   cout<<"Sum of 3 rows: "<<sumRow<<endl;

   for(int i = 0; i<3; i++){
      sumCol = 0;
      for(int j = 0; j<3; j++){
        sumCol = sumCol + a[j][i];
      }
   }
     cout<<"Sum of 3 columns: "<<sumCol;
   return 0;
}
```

C:\Users\hp\Documents\Sum of rows and columns.exe

```
Sum of 3 rows: 24
Sum of 3 columns: 18
--------------------------------
Process exited after 0.4292 seconds with return value 0
Press any key to continue . . .
```

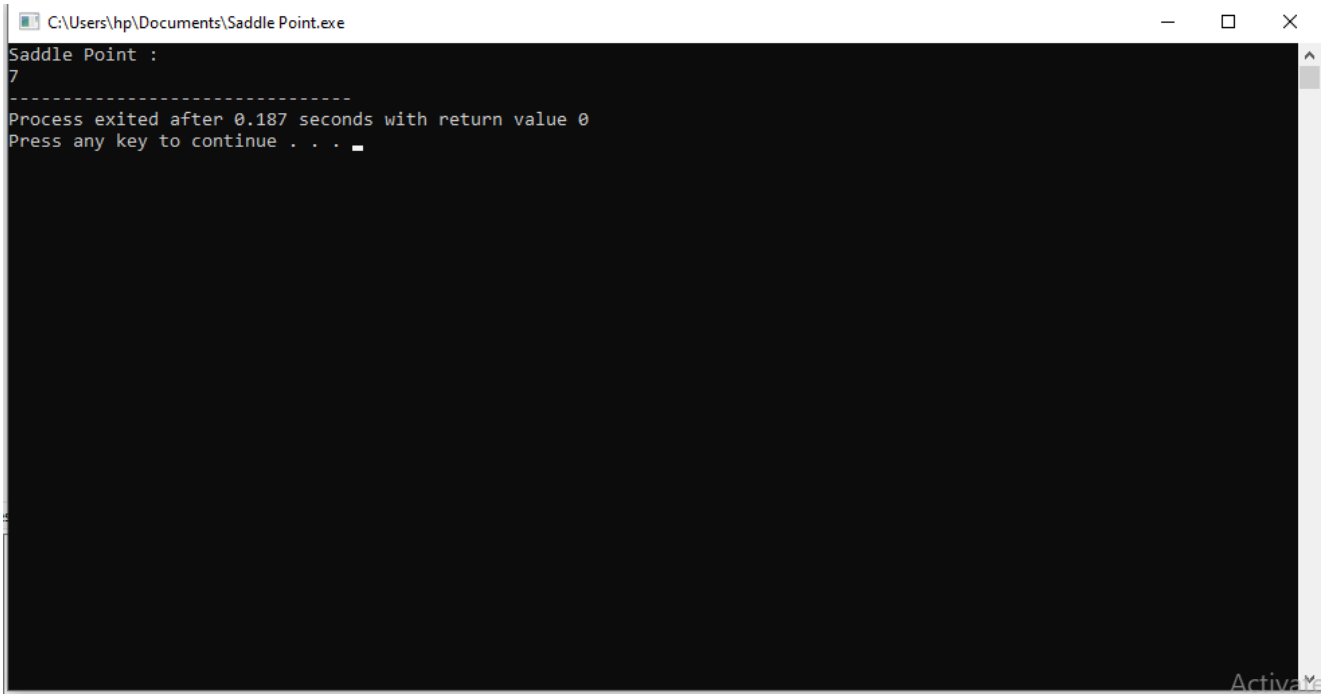
Question 4-

```
#include<iostream>
using namespace std;

int main()
{
   int n=3;
   int matrix[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
   cout<<"Saddle Point : "<<endl;
   int minRow[]={100001, -1};
          int maxCol=0;
      for(int i=0; i<n; i++){
         for(int j=0; j<n; j++){
            if(minRow[0]>matrix[i][j]){
               minRow[0]=matrix[i][j];
               minRow[1]=j;
            }

         }
         for(int k=0; k<n; k++){
            if(maxCol<matrix[k][minRow[1]]){
               maxCol=matrix[k][minRow[1]];
            }
         }
         if(minRow[0]==maxCol){
            cout<<maxCol<<" ";
```

```
        }
        minRow[0]=100001;
        maxCol=0;
    }
    return 0;
}
```



Question 5-

```
#include <iostream>
using namespace std;
int main()
{
	int n,m;
	cout << "Enter the number of rows : ";
	cin >> m;
	cout << "Enter the number of columns : ";
	cin >> n;
	int arr[m][n];
	int i,j;
	cout << "\nInput the matrix\n";
	for(i = 0; i < m; i++)
	{
		for(j = 0; j < n; j++)
		{
			cin >> arr[i][j];
		}
	}
	cout << "\nMatrix is\n";
	for(i = 0; i < m; i++)
	{
		for(j = 0; j < n; j++)
		{
			cout << arr[i][j] <<"\t";
```

```
        }
        cout << endl;
    }
    cout << "\nSpiral Matrix : ";
    int k = 0, l = 0;
    while(k < m && l < n)
    {
        for(i = l; i < n; i++)
        {
            cout << arr[k][i] <<" ";
        }
        k++;
        for(i = k; i < m; i++)
        {
            cout << arr[i][n-1] <<" ";
        }
        n--;
        if(k < m)
        {
            for(i = n - 1; i >= 0; --i)
            {
                cout << arr[m-1][i] <<" ";
            }
            m--;
        }
        if(l < n)
```

```
            {
                for(i = m - 1; i >= k; i--)
                {
                    cout << arr[i][l] <<" ";
                }
                l++;
            }
        }
        cout << endl;
        return 0;
}
```

Question 6-

```
#include<iostream>

#include<vector>


using namespace std;


vector<vector<int> > generate_spiral_matrix(int n)

{

        //Declaration of 2D vector.

   vector<vector<int> > result_matrix(n,vector<int>(n,0));


        // Normal Case

        int rowStart = 0;

        int rowEnd = n-1;

        int colStart = 0;

        int colEnd = n-1;

        int num = 1;


        while (rowStart <= rowEnd && colStart <= colEnd)

        {

                for (int i = colStart; i <= colEnd; i ++) // 1. horizonal, left to right

                {

                        result_matrix[rowStart][i] = num ++;

                }

                rowStart ++;
```

```
                for (int i = rowStart; i <= rowEnd; i ++) // 2. vertical, top to bottom
                {
                        result_matrix[i][colEnd] = num ++;
                }
                colEnd --;

                for (int i = colEnd; i >= colStart; i --)  // 3. horizonal, right to left
                {
                        if (rowStart <= rowEnd)
                                result_matrix[rowEnd][i] = num ++;
                }
                rowEnd --;

                for (int i = rowEnd; i >= rowStart; i --)  // 4. vertical, bottom to  top
                {
                        if (colStart <= colEnd)
            result_matrix[i][colStart] = num ++;
        }
          colStart ++;
    }

        return result_matrix;
}

int main()
{
```

```
        int n = 3;

        //Declare a 2d vector to get the result
        vector<vector<int> > result_matrix = generate_spiral_matrix(n);

        cout<< "Result matrix is : "<<endl;
        for (int i = 0; i < n; i++)
        {
                for (int j = 0; j < n; j++)
                        cout << result_matrix[i][j] <<"\t";
                        cout << endl;
        }
}
```