CS & Programming Lab
Lab Manual 09

Name: Abdullah Jamil Tung

Class: ME-15-Sec-A

Roll #: 478459

Lab Task #1

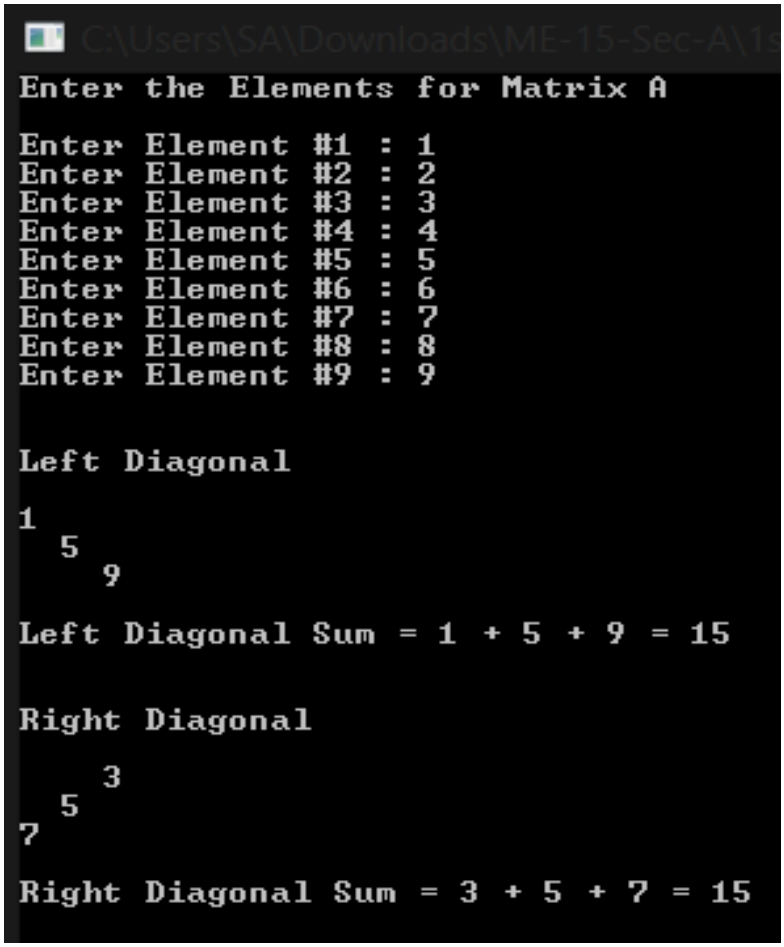
/*1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.*/

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

int main()
{
	//Declaring variables
	int a[3][3] , sum, x=1;
	//Taking Inputs
	cout<<"Enter the Elements for Matrix A\n\n";
	for (int i=0; i<3; i++)
	{
	for (int j=0; j<3; j++)
	{
	cout<<"Enter Element #"<<x<<" : ";
	cin>>a[i][j];
	x++;
```

```
}
}
cout<<"\n\n";

//Sum of Left Diagonal
cout<<"Left Diagonal\n\n";
for (int i=0; i<3; i++)
{
for (int j=0; j<3; j++)
{
if (i==j)
cout<<a[i][j]<<" ";
else
cout<<"  ";
}
cout<<endl;
}
cout<<endl;
sum = a[0][0] + a[1][1] + a[2][2];
cout<<"Left Diagonal Sum = "<<a[0][0]<<" + "<<a[1][1]<<" + "<<a[2][2]<<" = "<<sum<<"\n\n\n";
```

```
	//Sum of Right Diagonal
	cout<<"Right Diagonal\n\n";
	for (int i=0; i<3; i++)
	{
	for (int j=0; j<3; j++)
	{
	if (i==0&&j==2||i==1&&j==1||i==2&&j==0)
	cout<<a[i][j]<<" ";
	else
	cout<<"  ";
	}
	cout<<endl;
	}
	cout<<endl;
	sum = a[0][2] + a[1][1] + a[2][0];
	cout<<"Right Diagonal Sum = "<<a[0][2]<<" + "<<a[1][1]<<" +
"<<a[2][0]<<" = "<<sum<<"\n\n";
}
```

```cpp
/*1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix.*/

#include<iostream>
using namespace std;

int main()
{
    //Declaring variables
    int a[3][3] , sum, x=1;
    //Taking Inputs
    cout<<"Enter the Elements for Matrix A\n\n";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    cout<<"Enter Element #"<<x<<" : ";
    cin>>a[i][j];
    x++;
    }
    }
    cout<<"\n\n";

    //Sum of Left Diagonal
    cout<<"Left Diagonal\n\n";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    if (i==j)
    cout<<a[i][j]<<" ";
    else
    cout<<"  ";
    }
    cout<<endl;
    }
    cout<<endl;

    sum = a[0][0] + a[1][1] + a[2][2];
    cout<<"Left Diagonal Sum = "<<a[0][0]<<" + "<<a[1][1]<<" + "<<a[2][2]<<" = "<<sum<<"\n\n\n";

    //Sum of Right Diagonal
    cout<<"Right Diagonal\n\n";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    if (i==0&&j==2||i==1&&j==1||i==2&&j==0)
    cout<<a[i][j]<<" ";
    else
    cout<<"  ";
    }
    cout<<endl;
    }
    cout<<endl;
    sum = a[0][2] + a[1][1] + a[2][0];
    cout<<"Right Diagonal Sum = "<<a[0][2]<<" + "<<a[1][1]<<" + "<<a[2][0]<<" = "<<sum<<"\n\n";
}
```


C:\Users\SA\Downloads\ME-15-Sec-A\1
Enter the Elements for Matrix A
Enter Element #1 : 1
Enter Element #2 : 2
Enter Element #3 : 3
Enter Element #4 : 4
Enter Element #5 : 5
Enter Element #6 : 6
Enter Element #7 : 7
Enter Element #8 : 8
Enter Element #9 : 9
Left Diagonal
1
5
9
Left Diagonal Sum = 1 + 5 + 9 = 15
Right Diagonal
3
5
7
Right Diagonal Sum = 3 + 5 + 7 = 15

Lab Task #2

/*2. Write a function to add two 2D arrays of size 3x3.*/

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

//Defining function
int sum(int a[3][3], int b[3][3], int c[3][3])
{
	//Computing Result
	for (int i=0; i<3; i++)
	{
	for (int j=0; j<3; j++)
	{
	c[i][j] = a[i][j] + b[i][j];
	}}
	//Displaying Output
	cout<<"Sum of 2D Matrices =\t";
```

```
        for (int i=0; i<3; i++)
        {
        for (int j=0; j<3; j++)
        {
        cout<<c[i][j]<<" ";
        }
        cout<<endl<<"\t\t\t";
        }
}

//Main function
int main()
{
        //Declaring variables
        int a[3][3] = {{1,2,3} , {4,5,6} , {7,8,9}};
        int b[3][3] = {{1,2,3} , {4,5,6} , {7,8,9}};
        int c[3][3];
        //Calling function
        sum( a, b, c);
}
```

```cpp
/*2. Write a function to add two 2D arrays of size 3x3.*/

#include<iostream>
using namespace std;

//Defining function
int sum(int a[3][3], int b[3][3], int c[3][3])
{
    //Computing Result
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    c[i][j] = a[i][j] + b[i][j];
    }}
    //Displaying Output
    cout<<"Sum of 2D Matrices =\t";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    cout<<c[i][j]<<" ";
    }
    cout<<endl<<"\t\t\t";
    }
}

//Main function
int main()
{
    //Declaring variables
    int a[3][3] = {{1,2,3} , {4,5,6} , {7,8,9}};
    int b[3][3] = {{1,2,3} , {4,5,6} , {7,8,9}};
    int c[3][3];
    //Calling function
    sum( a, b, c);
}
```

```
C:\Users\SA\Downloads\ME-15-Sec-A\
Sum of 2D Matrices =    2 4 6
                        8 10 12
                        14 16 18

--------------------------------
```

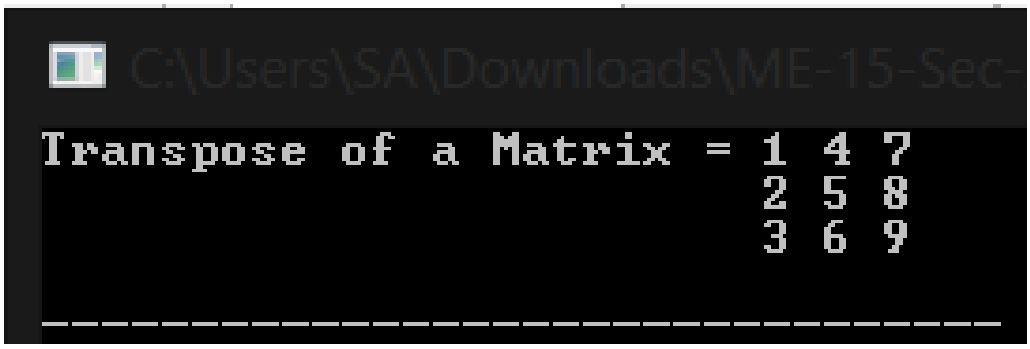
Lab Task #3

/*3. Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.*/

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

//Defining function
int transpose (int a[3][3], int b[3][3])
{
        //Computing result
        for (int i=0; i<3; i++)
        {
        for (int j=0; j<3; j++)
        {
        b[j][i]=a[i][j];
   }
        }
}
```

```
//Main function
int main()
{
	//Declaring variables
	int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, Transpose[3][3];
	//Calling function
	transpose (a,Transpose);
	//Displaying output
	cout<<"Transpose of a Matrix =\t";
	for (int i=0; i<3; i++)
	{
	for (int j=0; j<3; j++)
	{
	cout<<Transpose[i][j]<<" ";
	}
	cout<<endl<<"\t\t\t";
	}
}
```

```cpp
/*3. Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function.*/

#include<iostream>
using namespace std;

//Defining function
int transpose (int a[3][3], int b[3][3])
{
    //Computing result
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    b[j][i]=a[i][j];
    }
    }
}

//Main function
int main()
{
    //Declaring variables
    int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}}, Transpose[3][3];
    //Calling function
    transpose (a,Transpose);
    //Displaying output
    cout<<"Transpose of a Matrix =\t";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    cout<<Transpose[i][j]<<" ";
    }
    cout<<endl<<"\t\t\t";
    }
}
```



Lab Task #4

/*4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.*/

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

//Defining function
int Multiply(int a[3][3], int b[3][3], int Multi[3][3])
{
	//Computing result
	for (int i=0; i<3; i++)
	{
	for (int j=0; j<3; j++)
	{
	Multi[i][j]=0;
	for (int k=0; k<3; k++)
	{
	Multi[i][j]+=a[i][k]*b[k][j];
	}
```

```
        }
        }
}
//Main function
int main()
{
        //Declaring variables
        int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};
        int b[3][3]={{1,2,3},{4,5,6},{7,8,9}};
        int c[3][3];
        //Calling function
        Multiply(a,b,c);
        //Displaying output
        cout<<"Multiplication of Two 3x3 Matrices =\t";
        for (int i=0; i<3; i++)
        {
        for (int j=0; j<3; j++)
        {
        cout<<c[i][j]<<" ";
        }
        cout<<endl<<"\t\t\t\t\t";
```

```
    }
}
```

```cpp
/*4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.*/

#include<iostream>
using namespace std;

//Defining function
int Multiply(int a[3][3], int b[3][3], int Multi[3][3])
{
    //Computing result
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    Multi[i][j]=0;
    for (int k=0; k<3; k++)
    {
    Multi[i][j]+=a[i][k]*b[k][j];
    }
    }
    }
}

//Main function
int main()
{
    //Declaring variables
    int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    int b[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    int c[3][3];
    //Calling function
    Multiply(a,b,c);
    //Displaying output
    cout<<"Multiplication of Two 3x3 Matrices =\t";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    cout<<c[i][j]<<" ";
    }
    cout<<endl<<"\t\t\t\t\t";
    }
}
```

```
C:\Users\SA\Downloads\ME-15-Sec-A\1st Semester\Courses
Multiplication of Two 3x3 Matrices =    30 36 42
                                        66 81 96
                                        102 126 150

--------------------------------
```

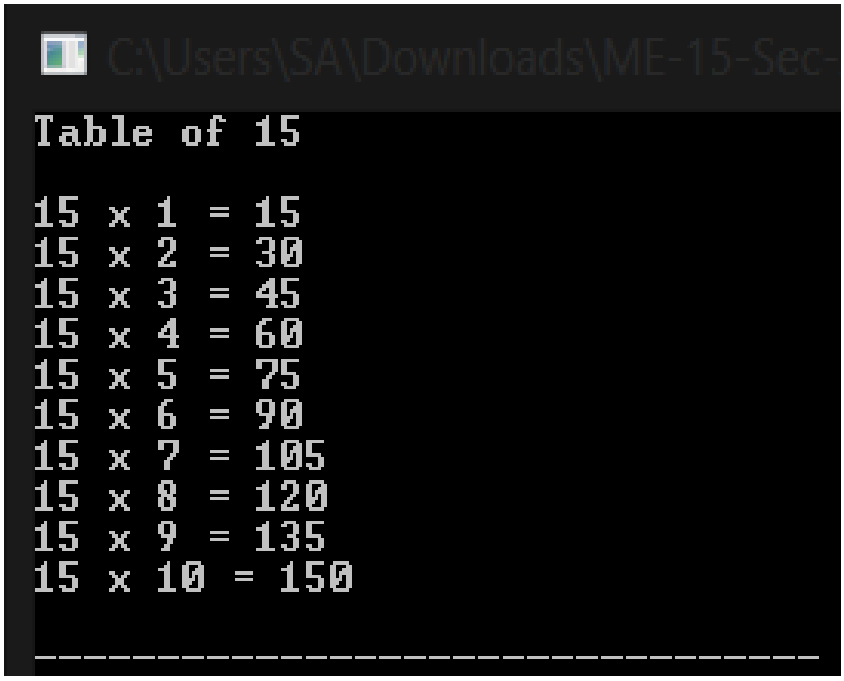
Lab Task #5

/*5. Print the multiplication table of 15 using recursion.*/

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

//Making a recursive function
int Table_Of_15(int start, int end)
{
	//Checking for invalid operations
	if (start<=0)
	return 0;

	//Computing result
	else if (start<=end)
	{cout<<"15 x "<<start<<" = "<<15*start<<endl;
	//Recalling function
	return Table_Of_15(start+1, end);}
}
```

```
//Main Function
int main()
{
	//Declaring variables
	int x=1, y=10;
	//Displaying output
	cout<<"Table of 15\n\n";
	Table_Of_15(x,y);
}
```

```
/*5. Print the multiplication table of 15 using recursion.*/

#include<iostream>
using namespace std;

//Making a recursive function
int Table_Of_15(int start, int end)
{
    //Checking for invalid operations
    if (start<=0)
    return 0;

    //Computing result
    else if (start<=end)
    {cout<<"15 x "<<start<<" = "<<15*start<<endl;
    //Recalling function
    return Table_Of_15(start+1, end);}
}

//Main Function
int main()
{
    //Declaring variables
    int x=1, y=10;
    //Displaying output
    cout<<"Table of 15\n\n";
    Table_Of_15(x,y);
}
```


C:\Users\SA\Downloads\ME-15-Sec-
Table of 15
15 x 1 = 15
15 x 2 = 30
15 x 3 = 45
15 x 4 = 60
15 x 5 = 75
15 x 6 = 90
15 x 7 = 105
15 x 8 = 120
15 x 9 = 135
15 x 10 = 150

Home Task #1

/*1. Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.*/

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

int main()
{
	//Declaring variables
	float m[3][3], determ, det_1, det_2, det_3, cofac[3][3], adj[3][3], inv[3][3], x=1;

	//Taking inputs
	cout<<"Enter the Elements for the 3x3 Matrix A\n"<<endl;
	for (int i=0; i<3; i++)
	{
	for (int j=0; j<3; j++)
	{
	cout<<"Enter Element #"<<x<<" = ";
```

```
    cin>>m[i][j];
    x++;
    }
    }
cout<<"\n\n";

    //Calculating Determinant
det_1 = m[0][0] * (m[1][1]*m[2][2] - m[2][1]*m[1][2]);
det_2 = -m[0][1] * (m[1][0]*m[2][2] - m[2][0]*m[1][2]);
    det_3 = m[0][2] * (m[1][0]*m[2][1] - m[2][0]*m[1][1]);
    determ = det_1 + det_2 + det_3;
    if (determ == 0)
    cout<<"Matrix cannot be inverted."<<endl;

    else
    {//Calculating Cofactor matrix
    cofac[0][0] = m[1][1]*m[2][2] - m[2][1]*m[1][2];
cofac[0][1] = -(m[1][0]*m[2][2] - m[2][0]*m[1][2]);
cofac[0][2] = m[1][0]*m[2][1] - m[2][0]*m[1][1];
cofac[1][0] = -(m[0][1]*m[2][2] - m[2][1]*m[0][2]);
cofac[1][1] = m[0][0]*m[2][2] - m[2][0]*m[0][2];
```

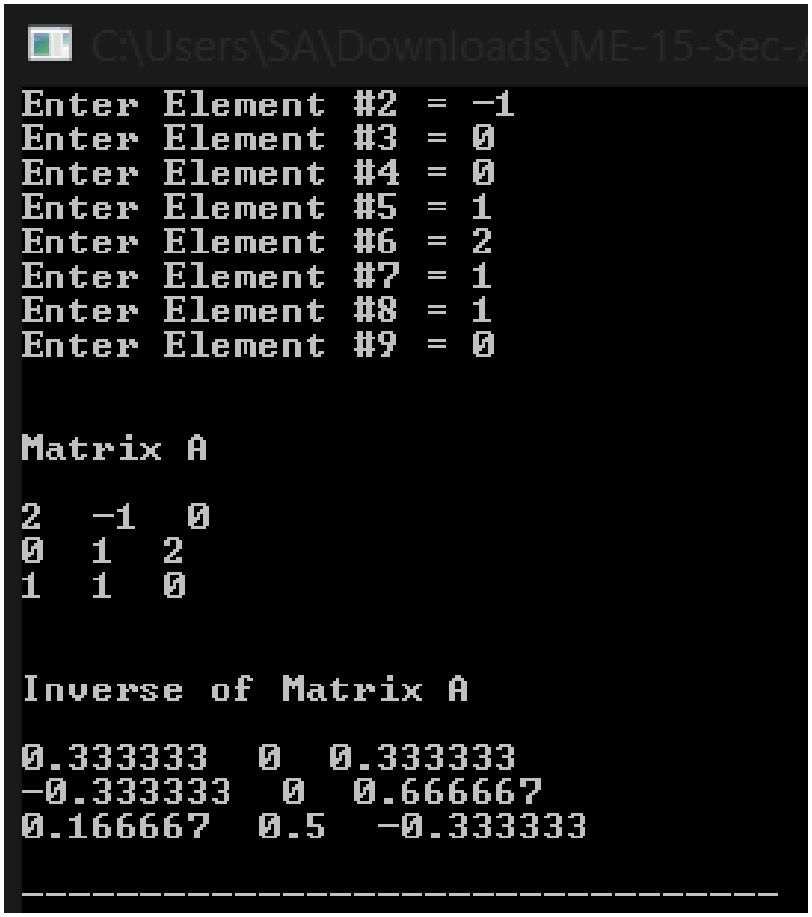
```
cofac[1][2] = -(m[0][0]*m[2][1] - m[2][0]*m[0][1]);
cofac[2][0] = m[0][1]*m[1][2] - m[1][1]*m[0][2];
cofac[2][1] = -(m[0][0]*m[1][2] - m[1][0]*m[0][2]);
cofac[2][2] = m[0][0]*m[1][1] - m[1][0]*m[0][1];

//Calculating Adjoint
for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    adj[j][i]=cofac[i][j];
}
    }

    //Displaying Matrix A
    cout<<"Matrix A\n\n";
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    cout<<m[i][j]<<"  ";
```

```
        }
        cout<<endl;
        }
        cout<<"\n\n";

        //Displaying Inverse of Matrix A
        cout<<"Inverse of Matrix A\n\n";
        for (int i=0; i<3; i++)
        {
        for (int j=0; j<3; j++)
        {
        //Computing Inverse of Matrix A
        inv[i][j] = adj[i][j]/determ;
        if (inv[i][j]==-0)
        inv[i][j]=0;
        cout<<inv[i][j]<<"  ";
        }
        cout<<endl;
        }
        }
}
```

```cpp
/*1. Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint.*/

#include<iostream>
using namespace std;

int main()
{
    //Declaring variables
    float m[3][3], determ, det_1, det_2, det_3, cofac[3][3], adj[3][3], inv[3][3], x=1;

    //Taking inputs
    cout<<"Enter the Elements for the 3x3 Matrix A\n"<<endl;
    for (int i=0; i<3; i++)
    {
    for (int j=0; j<3; j++)
    {
    cout<<"Enter Element #"<<x<<" = ";
    cin>>m[i][j];
    x++;
    }
    }
    cout<<"\n\n";

    //Calculating Determinant
    det_1 = m[0][0] * (m[1][1]*m[2][2] - m[2][1]*m[1][2]);
    det_2 = -m[0][1] * (m[1][0]*m[2][2] - m[2][0]*m[1][2]);
    det_3 = m[0][2] * (m[1][0]*m[2][1] - m[2][0]*m[1][1]);
    determ = det_1 + det_2 + det_3;
    if (determ == 0)
    cout<<"Matrix cannot be inverted."<<endl;

```

```
32      else
33      {//Calculating Cofactor matrix
34      cofac[0][0] = m[1][1]*m[2][2] - m[2][1]*m[1][2];
35      cofac[0][1] = -(m[1][0]*m[2][2] - m[2][0]*m[1][2]);
36      cofac[0][2] = m[1][0]*m[2][1] - m[2][0]*m[1][1];
37      cofac[1][0] = -(m[0][1]*m[2][2] - m[2][1]*m[0][2]);
38      cofac[1][1] = m[0][0]*m[2][2] - m[2][0]*m[0][2];
39      cofac[1][2] = -(m[0][0]*m[2][1] - m[2][0]*m[0][1]);
40      cofac[2][0] = m[0][1]*m[1][2] - m[1][1]*m[0][2];
41      cofac[2][1] = -(m[0][0]*m[1][2] - m[1][0]*m[0][2]);
42      cofac[2][2] = m[0][0]*m[1][1] - m[1][0]*m[0][1];
43
44      //Calculating Adjoint
45      for (int i=0; i<3; i++)
46      {
47      for (int j=0; j<3; j++)
48      {
49      adj[j][i]=cofac[i][j];
50      }
51      }
52
53      //Displaying Matrix A
54      cout<<"Matrix A\n\n";
55      for (int i=0; i<3; i++)
56      {
57      for (int j=0; j<3; j++)
58      {
59      cout<<m[i][j]<<"  ";
60      }
61      cout<<endl;
62      }
63      cout<<"\n\n";
64
```

```
65      //Displaying Inverse of Matrix A
66      cout<<"Inverse of Matrix A\n\n";
67      for (int i=0; i<3; i++)
68      {
69      for (int j=0; j<3; j++)
70      {
71      //Computing Inverse of Matrix A
72      inv[i][j] = adj[i][j]/determ;
73      if (inv[i][j]==-0)
74      inv[i][j]=0;
75      cout<<inv[i][j]<<"  ";
76      }
77      cout<<endl;
78      }
79      }
80  }
```


C:\Users\SA\Downloads\ME-15-Sec-
Enter Element #2 = -1
Enter Element #3 = 0
Enter Element #4 = 0
Enter Element #5 = 1
Enter Element #6 = 2
Enter Element #7 = 1
Enter Element #8 = 1
Enter Element #9 = 0
Matrix A
2 -1 0
0 1 2
1 1 0
Inverse of Matrix A
0.333333 0 0.333333
-0.333333 0 0.666667
0.166667 0.5 -0.333333