Cramer’s Rule

Mohamed Soliman 900182638
Mohamed Bedda 900161918
Omar Fayed 900191831

Part 2 (40 Marks):
A matlab code Write a programming code using MATLAB, Python, C++ or any appropriate language to solve a system of n linear equations in n variables using Cramer’s Rule.

Our c++ code:

```
#include<iostream>
#include<math.h>
#include <Eigen/LU>
#include <iomanip>
#include <vector>

using namespace Eigen;
using namespace std;

MatrixXd ConvertToEigenMatrix(vector<vector<double>> data)
{
   MatrixXd eMatrix(data.size(), data[0].size());
   for (int i = 0; i < data.size(); ++i)
      eMatrix.row(i) = VectorXd::Map(&data[i][0], data[0].size());
   return eMatrix;
}

void determinant(MatrixXd mat) {


   MatrixXd inverse = mat.inverse();
   double determinant = mat.determinant();

   cout << "Its determinant is " << determinant << endl;
   if (determinant != 0) {

      cout << "It is invertible, and its inverse is:" << endl << inverse << endl;
   }
```

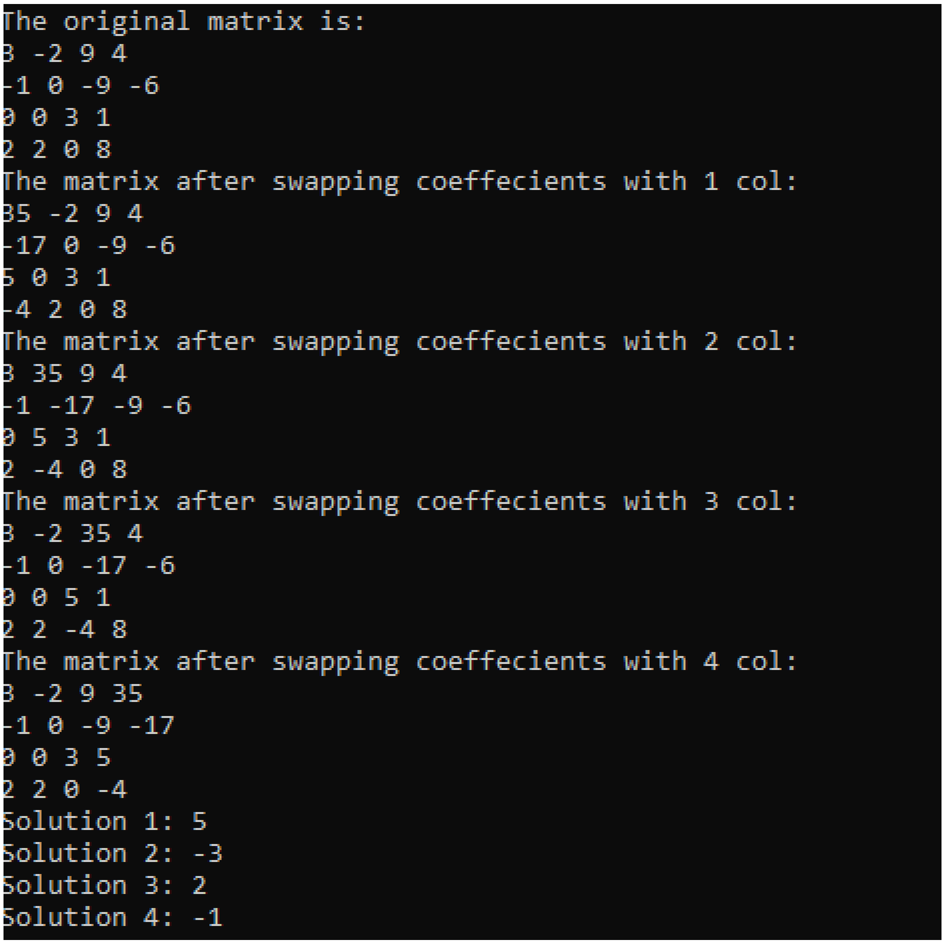
```
    else {
        cout << "It is not invertible." << endl;
    }

}
int matrix_size;

void getCofactor(int** mat, int** temp, int p, int q, int n);

void Carmer(int** mat, int* coefficients);
int main()
{
    int user;

    cout << "Enter 1 to input matrix or 2 to load predefined matrix.\n";
    cin >> user;

    if (user == 1) {

        int n, d, i, j;


        cout << "Enter the size of the matrix:\n";

        cin >> n;

        matrix_size = n;

        vector<vector<double>> matrix(n);


        cout << "Enter the coefficients of the matrix:\n";
        vector<double> coeff;
        for (i = 0; i < n; i++) {
            int input;
            cin >> input;
            coeff.push_back(input);
        }
        cout << endl;

        int* coefficients = new int[n];
        for (int i = 0, k = 0; i < n; i++, k++) {
            coefficients[i] = coeff[k];
        }
```

```
cout << "Enter the elements of the matrix in the following format:\n\n";

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

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

        cout << "m" << (i + 1) << (j + 1) << " ";

    cout << endl;

}


cout << endl;

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

    matrix[i] = vector<double>(n);

    for (int j = 0; j < n; j++)
    {
        cin >> matrix[i][j];
    }
}

int** mat2 = new int* [n];
for (int i = 0; i < n; i++) {
    mat2[i] = new int[n];
}

for (int i = 0, k = 0; i < n; i++) {
    for (int j = 0; j < n; j++, k++) {
        mat2[i][j] = matrix[i][j];
    }
}

cout << endl;

cout << "The entered matrix is:" << endl;

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

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

```
            cout << matrix[i][j] << " ";

        cout << endl;
    }

    MatrixXd mat = ConvertToEigenMatrix(matrix);

    determinant(mat);

    Carmer(mat2, coefficients);

}
else if (user == 2) {

    matrix_size = 4
    int** mat = new int* [matrix_size];
    for (int i = 0; i < matrix_size; i++) {
        mat[i] = new int[matrix_size];
    }
    int a[] = { 3,-2,9,4,-1,0,-9,-6,0,0,3,1,2,2,0,8 };
    int b[] = { 35,-17,5,-4 };
    int* coefficients = new int[matrix_size];
    for (int i = 0, k = 0; i < matrix_size; i++, k++) {
        coefficients[i] = b[k];
    }
    for (int i = 0, k = 0; i < matrix_size; i++) {
        for (int j = 0; j < matrix_size; j++, k++) {
            mat[i][j] = a[k];
        }
    }
    for (int i = 0; i < matrix_size; i++) {
        for (int j = 0; j < matrix_size; j++) {
            cout << mat[i][j] << " ";
        }
        cout << endl;
    }
    Carmer(mat, coefficients);

}
else {
    cout << "Invalid entry. Exiting Program...\n";
}
```

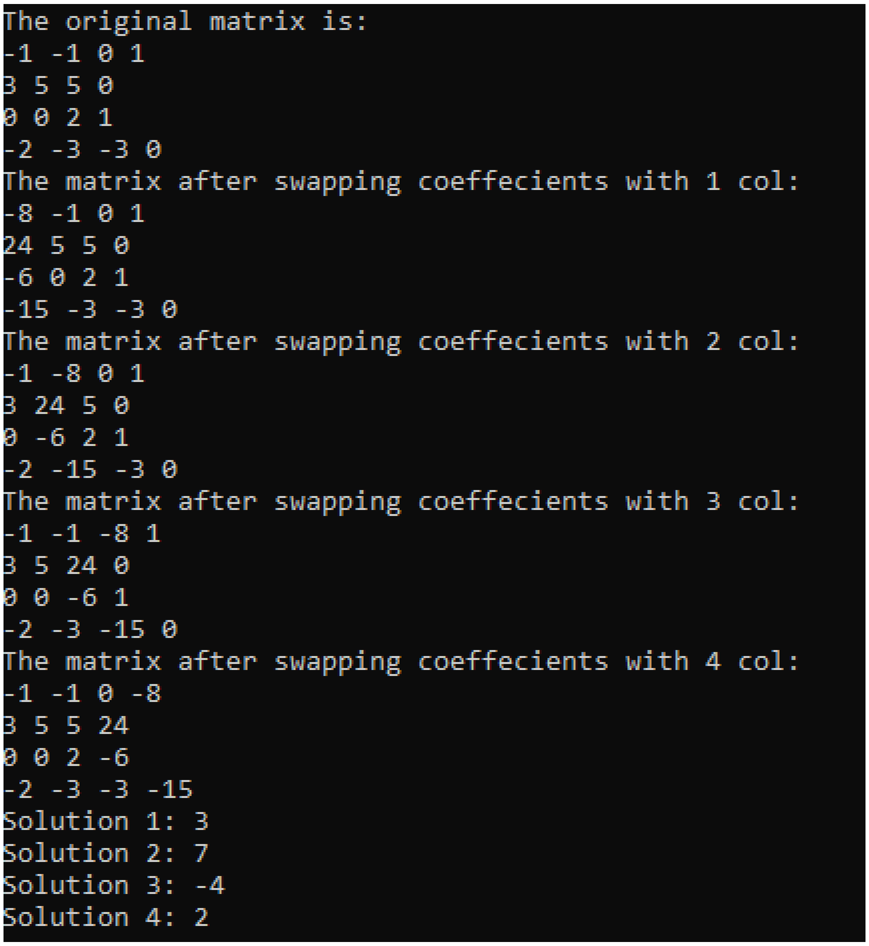
```
    return 0;
}
void getCofactor(int** mat, int** temp, int p, int q, int n)
{
    int i = 0, j = 0;
    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n; col++) {
            if (row != p && col != q) {
                temp[i][j++] = mat[row][col];
                if (j == n - 1) {
                    j = 0;
                    i++;
                }
            }
        }
    }
}
int determinantOfMatrix(int** mat, int n)
{
    int D = 0;
    if (n == 1)
        return mat[0][0];
    int** temp = new int* [matrix_size];
    for (int i = 0; i < matrix_size; i++) {

        temp[i] = new int[matrix_size];
    }
    int sign = 1;
    for (int f = 0; f < n; f++) {
        getCofactor(mat, temp, 0, f, n);
        D += sign * mat[0][f] * determinantOfMatrix(temp, n - 1);
        sign = -sign;
    }

    return D;
}


void Carmer(int** mat, int* coefficients) {
    int** temp = new int* [matrix_size];
    vector<double> determinants;

    for (int i = 0; i < matrix_size; i++) {
        temp[i] = new int[matrix_size];
    }
    for (int i = 0; i < matrix_size; i++) {
        for (int j = 0; j < matrix_size; j++) {
            temp[i][j] = mat[i][j];
        }
    }
    determinants.push_back(determinantOfMatrix(temp, matrix_size));
    for (int k = 0, m = 0; k < matrix_size; k++) {
        for (int i = 0; i < matrix_size; i++) {
            for (int j = 0; j < matrix_size; j++) {
                temp[i][j] = mat[i][j];
            }
        }
        for (int i = 0; i < matrix_size; i++) {
            temp[i][k] = coefficients[i];
        }
        determinants.push_back(determinantOfMatrix(temp, matrix_size));

    }
    cout << "Using Cramer's Rule, the following is achieved:\n";

    if (determinants[0] != 0) {
        for (int i = 1; i < determinants.size(); i++) {
            double value = determinants[i] / determinants[0];
            cout << "Solution " << i << ": " << value << endl;
        }
    }
    else {
        int m = 0;
        for (int i = 1; i < determinants.size(); i++) {
            if (determinants[i] == 0)
                m++;
        }
        if (m == 3)
            cout << "Infinite solutions\n";
        else if (m == 0)
            cout << "No solutions\n";
    }
}
```

Inputs/Outputs of problem 25 of Section 3.4, page 142.



```
The original matrix is:
3 -2 9 4
-1 0 -9 -6
0 0 3 1
2 2 0 8
The matrix after swapping coeffecients with 1 col:
35 -2 9 4
-17 0 -9 -6
5 0 3 1
-4 2 0 8
The matrix after swapping coeffecients with 2 col:
3 35 9 4
-1 -17 -9 -6
0 5 3 1
2 -4 0 8
The matrix after swapping coeffecients with 3 col:
3 -2 35 4
-1 0 -17 -6
0 0 5 1
2 2 -4 8
The matrix after swapping coeffecients with 4 col:
3 -2 9 35
-1 0 -9 -17
0 0 3 5
2 2 0 -4
Solution 1: 5
Solution 2: -3
Solution 3: 2
Solution 4: -1
```

Inputs/Outputs of problem 26 of Section 3.4, page 142.



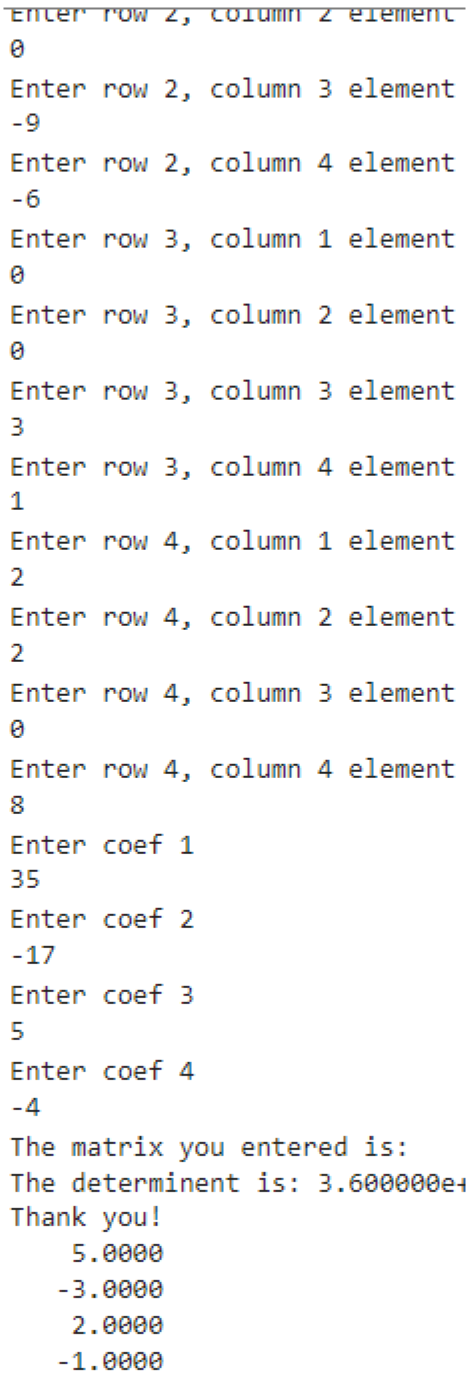
```
The original matrix is:
-1 -1 0 1
3 5 5 0
0 0 2 1
-2 -3 -3 0
The matrix after swapping coeffecients with 1 col:
-8 -1 0 1
24 5 5 0
-6 0 2 1
-15 -3 -3 0
The matrix after swapping coeffecients with 2 col:
-1 -8 0 1
3 24 5 0
0 -6 2 1
-2 -15 -3 0
The matrix after swapping coeffecients with 3 col:
-1 -1 -8 1
3 5 24 0
0 0 -6 1
-2 -3 -15 0
The matrix after swapping coeffecients with 4 col:
-1 -1 0 -8
3 5 5 24
0 0 2 -6
-2 -3 -3 -15
Solution 1: 3
Solution 2: 7
Solution 3: -4
Solution 4: 2
```

Our matlab code:

```
a = input('Enter the size of the matrix:');
b = a;
for i=1:a
   for j=1:b
      m(i,j)=input(sprintf('Enter row %d, column %d element',i,j));
   end
end
for i=1:a
   c(i)=input(sprintf('Enter coef %d',i));
end
disp('The matrix you entered is: ')
m=reshape(m,a,b);
d=det(m);
fprintf('The determinent is: %d', d)
if d == 0
          disp('The matrix is not invertible.')
else
          disp('The matrix is invertible, and here is the inverse:')
```

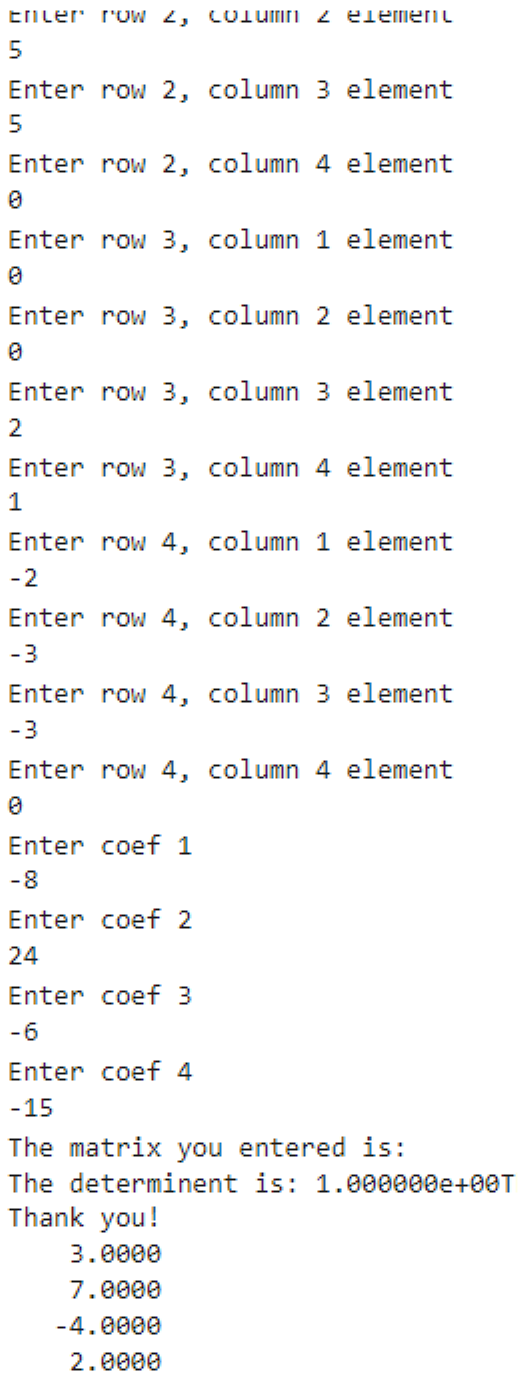
```
i=inv(m);
end
disp('Thank you!')
disp(Cramers_Rule(m, c));
%disp(Cramers_Rule(mat, coefficients));
function[answer] = Cramers_Rule(A,b)
   len = length(b);
   result = zeros(len,1);
   determinent = det(A);
   if A==0
       error('No Solution')
   elseif determinent==0
       error('No Solution')
   end
   for i=1:len
       Aug=A;
       Aug(:,i) = b;
       result(i) = (det(Aug)/determinent);
   end
   answer = result;
end
```

Inputs/Outputs of problem 25 of Section 3.4, page 142.



```
Enter row 2, column 2 element
0
Enter row 2, column 3 element
-9
Enter row 2, column 4 element
-6
Enter row 3, column 1 element
0
Enter row 3, column 2 element
0
Enter row 3, column 3 element
3
Enter row 3, column 4 element
1
Enter row 4, column 1 element
2
Enter row 4, column 2 element
2
Enter row 4, column 3 element
0
Enter row 4, column 4 element
8
Enter coef 1
35
Enter coef 2
-17
Enter coef 3
5
Enter coef 4
-4
The matrix you entered is:
The determinent is: 3.600000e+
Thank you!
    5.0000
   -3.0000
    2.0000
   -1.0000
```

Inputs/Outputs of problem 26 of Section 3.4, page 142.



```
Enter row 2, column 2 element
5
Enter row 2, column 3 element
5
Enter row 2, column 4 element
0
Enter row 3, column 1 element
0
Enter row 3, column 2 element
0
Enter row 3, column 3 element
2
Enter row 3, column 4 element
1
Enter row 4, column 1 element
-2
Enter row 4, column 2 element
-3
Enter row 4, column 3 element
-3
Enter row 4, column 4 element
0
Enter coef 1
-8
Enter coef 2
24
Enter coef 3
-6
Enter coef 4
-15
The matrix you entered is:
The determinent is: 1.000000e+00T
Thank you!
    3.0000
    7.0000
   -4.0000
    2.0000
```