**PROGRAM: 6**

**Name: Kiran Kanyal**

**Roll No: 29**

**Section: A1**

**Aim: Write a program in C language to find the solution for the given system of equations using the Gauss Elimination method.**

**ALGORITHM:**

**START**

1. Prompt the user to input the number of equations n.
2. Prompt the user to input the augmented A matrix of size n x (n+1).
3. For each row i from 1 to n-1:

For each row j below the current row (j=i+1 to n):

1. Calculate the multiplier mult = A[j][i]/A[i][i].
2. For each column k from i to n+1: Subtract mult\*A[i][k] from A[j][k].
3. If any pivot element is zero, check if the system has no solution or infinite solutions and stop if so.
4. For each row i from n to 1 (in reverse order):
   1. Solve for the variable x[i] using the equation:

x[i] = (A[i][n+1] - sum(A[i][j] \* x[j])) / A[i][i], where j ranges from i+1 to n.

1. Display the solutions x1, x2, ..., xn.

**STOP**

**PROGRAM:**

#include <stdio.h>

int main()

{

int n;

printf("Enter the order of matrix: ");

scanf("%d", &n);

float A[n][n + 1];

printf("Enter coefficients for x1, x2, ..., x%d and the constant terms:\n", n);

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

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

scanf("%f", &A[i][j]);

}

}

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

for (int j = i + 1; j < n; j++){

if (A[i][i] == 0){

printf("Unique solution does not exist.\n");

return -1;

}

for (int k = 0; k <= n; k++){

A[j][k] = A[j][k] - (A[j][i] / A[i][i]) \* A[i][k];

}

}

}

printf("Upper Triangular Matrix:\n");

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

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

printf("%.0f\t", A[i][j]);

}

printf("\n");

}

float x[n];

for (int i = n - 1; i >= 0; i--){

x[i] = A[i][n];

for (int j = i + 1; j < n; j++){

x[i] -= A[i][j] \* x[j];

}

x[i] /= A[i][i];

}

printf("The solution is : \n");

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

printf("x%d = %.3f\n", i + 1, x[i]);

}

return 0;

}

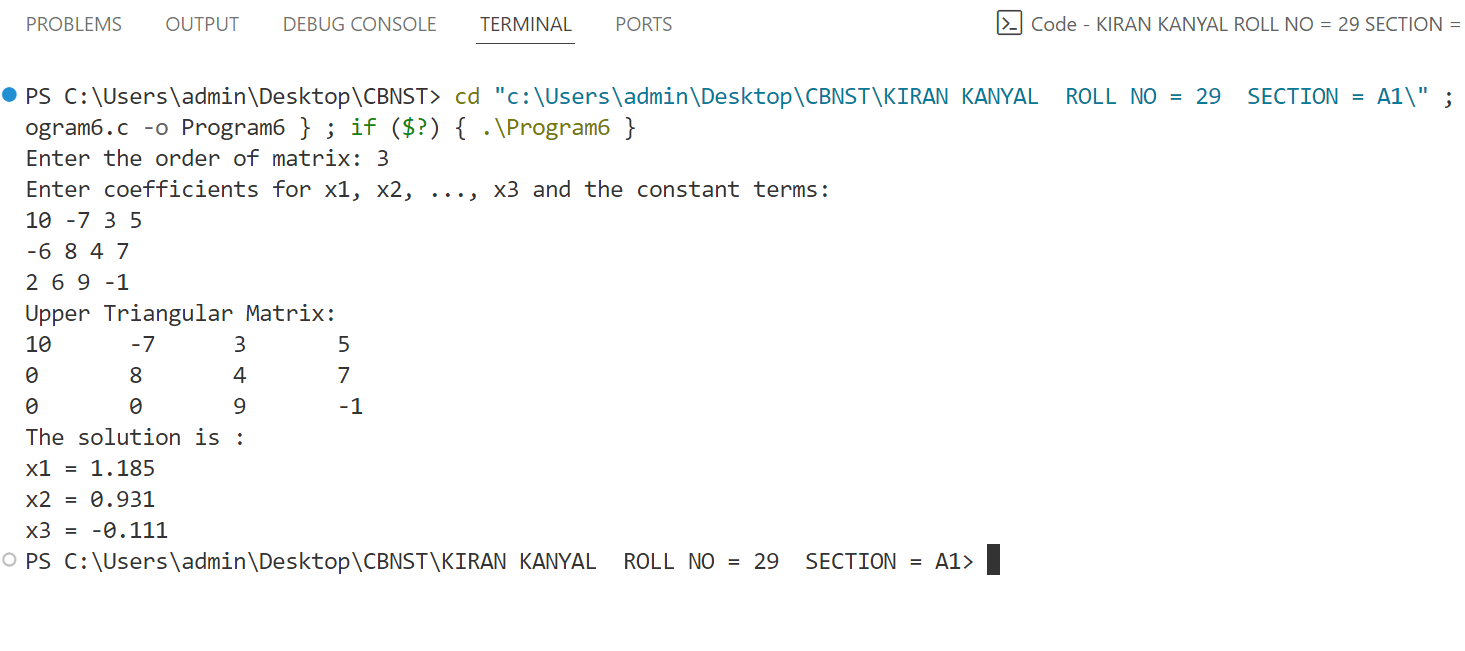
**OUTPUT:**

For the given system of equations:

10x1 -7x2 +3x3 = 5

-6x1 + 8x2 +4x3 =7

2x1 +6x2 +9x3 = -1



**PROGRAM: 7**

**Name: Kiran Kanyal**

**Roll No: 29**

**Section: A1**

**Aim: Write a program in C language to find the solution for the given system of equations using the Gauss-Jordan method.**

**ALGORITHM:**

**START**

1. Prompt the user to input the number of equations n.
2. Prompt the user to input the augmented matrix A of size n x (n+1).
3. For each row i from 1 to n:
   1. Normalize the pivot A[i][i] by dividing the entire row by A[i][i] to make the diagonal element 1.
   2. For each row j ≠ i:
4. Calculate the multiplier mult = A[j][i].

ii. For each column k from i to n+1, subtract mult \* A[i][k] from A[j][k] to make the element A[j][i] = 0.

1. If any diagonal element A[i][i] is zero during the process, check if the system has no solution or infinite solutions and stop if so.
2. The solutions x1, x2, ..., xn are now available directly in the last column of the augmented matrix.
3. Display the solutions x1, x2, ..., xn.

**STOP**

**PROGRAM:**

#include <stdio.h>

int main()

{

int n;

printf("Enter the order of matrix: ");

scanf("%d", &n);

float A[n][n + 1];

printf("Enter coefficients for x1, x2, ..., x%d and the constant terms:\n", n);

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

{

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

{

scanf("%f", &A[i][j]);

}

}

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

{

// Make the diagonal element 1

float diag = A[i][i];

if (diag == 0)

{

printf("Unique solution does not exist.\n");

return -1;

}

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

{

A[i][j] /= diag;

}

// Make all other elements in column i zero

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

{

if (j != i)

{

float factor = A[j][i];

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

{

A[j][k] -= factor \* A[i][k];

}

}

}

}

// Print the resulting identity matrix with constants

printf("Reduced Row Echelon Form (RREF):\n");

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

{

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

{

printf("%.1f\t", A[i][j]);

}

printf("\n");

}

printf("The solution is : \n");

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

{

printf("x%d = %.0f\n", i + 1, A[i][n]);

}

return 0;

}

**OUTPUT:**

For the given system of equations:

x1 + 4x2 + 9x3 = 16

2x1 + x2 + x3 = 10

3x1 + 2x2 + 3x3 = 18

