```c
#include<stdio.h>
#include<math.h>


void getOffsetToAdd(int a, int b, int c, int x, int *temp)
{
    int temp3 = c - a * (x - 10);
    int val1 = temp3 / b;
    temp3 = c - a * (x + 10);
    int val2 = temp3 / b;
    int x1 = (x - 10) < (x + 10) ? x - 10 : x + 10;
    int y1 = val1 < val2 ? val1 : val2;
    int offsetToAdd = 0;
    if (!(x1 >= 0 && y1 >= 0))
        offsetToAdd = x1 < y1 ? x1 : y1;
    x1 = (x - 10) > (x + 10) ? x - 10 : x + 10;
    y1 = val1 > val2 ? val1 : val2;
    offsetToAdd = abs(offsetToAdd);
    *temp = x1;
    *(temp + 1) = y1;
    *(temp + 2) = offsetToAdd;
}


void plotGraph(int a, int b, int c, int p, int q, int r, int x, int y)
{
    int temp[3];
    getOffsetToAdd(a, b, c, x, temp);
    int x1 = temp[0], y1 = temp[1], v1 = temp[2];
    getOffsetToAdd(p, q, r, x, temp);
    int x2 = temp[0], y2 = temp[1], v2 = temp[2];
    int offsetToAdd = v1 < v2 ? v2 : v1;

    int x3 = x1 < x2 ? x2 : x1;
    int y3 = y1 < y2 ? y2 : y1;
    int arr[x3 + offsetToAdd][y3 + offsetToAdd];
    int i, j;
    for (i = 0; i < x3 + offsetToAdd; i++)
    {
        for (j = 0; j < y3 + offsetToAdd; j++)
            arr[i][j] = 0;
    }

    for (i = x - 10; i < x + 10; i++)
    {
        int temp3 = c - a * i;
        y = temp3 / b;
        arr[i + offsetToAdd][y + offsetToAdd] = 1;
    }

    for (i = x - 10; i < x + 10; i++)
    {
        int temp3 = r - p * i;
        y = temp3 / q;
        arr[i + offsetToAdd][y + offsetToAdd] = 2;
    }
    printf("\n");
    for (i = 0; i < x3 + offsetToAdd; i++)
    {
        for (j = 0; j < y3 + offsetToAdd; j++)
```

```
            arr[i][j] == 0 ? printf("%c ", ' ') : printf("%d ", arr[i][j]);

        printf("\n");
    }
}


int compare_float(float f1, float f2)
{
    float precision = 0.00001;
    if (fabs(f1 - f2) < precision)
        return 1;
    return 0;
}


void substitution(int a, int b, int c, int p, int q, int r, float *result)
{
    printf("\n\nSolving using Substitution Method:\n\n");
    float x = (float)(b*r - q*c)/(p*b - a*q);
    printf("The calculated value of x is = %.2f\n", x);
    int temp3 = c - a * x;
    float y = (float)temp3 / b;
    printf("The calculated value of y is = %.2f\n", y);
    *result = x;
    *(result + 1) = y;
}


void matrixMultiplication(int a, int b, int c, int p, int q, int r, float *result)
{
    printf("\n\nSolving using Matrix Multiplication Method: \n\t[X = inverse(A)B]\n\n");
    int determinant = a * q - b * p;
    printf("The value of determinant = %d", determinant);
    printf("\nThe value of resultant A inverse is:\n");
    float adj[2][2] = {{q, -b}, {-p, a}};
    float inverse[2][2];
    int B[2][1] = {{c}, {r}};
    int i, j, k;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
            printf("%.2f\t", (float)adj[i][j] / determinant);
            inverse[i][j] = (float)adj[i][j] / determinant;
        }
        printf("\n");
    }
    printf("\nThe value of B is:\n");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 1; j++)
            printf("%d", B[i][j]);
        printf("\n");
    }
    float res[2][1];
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 2; j++)
        {
```

```
            res[i][j] = 0;
            for (k = 0; k < 2; k++)
                res[i][j] += inverse[i][k] * B[k][j];
        }
    }
    printf("The value of resultant matrix is:\n");
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 1; j++)
            printf("%.2f", res[i][j]);
        printf("\n");
    }
    printf("\nThe calculated value of x is = %.2f\n", res[0][0]);
    printf("The calculated value of y is = %.2f\n", res[1][0]);
    *result = res[0][0];
    *(result + 1) = res[1][0];
}


int main() {
    int a, b, c;
    int p, q, r;
    printf("The equations should be in the format of:\n\tax + by = c and\n\tpx + qy = r\n");
    printf("Enter a, b, c, p, q, r: ");
    scanf("%d %d %d %d %d %d" , &a, &b, &c, &p, &q, &r);

    char c1, c2;
    c1 = b < 0 ? atoi("45") : atoi("43");
    c2 = q < 0 ? atoi("45") : atoi("43");
    printf("The provided equations are:\n\t%dx %c %dy = %d\n\t%dx %c %dy = %d", a, c1, abs(b),

    float m1 = -(float)a / b;
    float m2 = -(float)p / q;
    if (compare_float(m1, m2))
    {
        printf("\n\n\nThe lines are parallel and solution doesn't exist");
        return 0;
    }

    float s_res[2], m_res[2];
    substitution(a, b, c, p, q ,r ,s_res);
    matrixMultiplication(a, b, c, p, q ,r ,m_res);
    if (compare_float(s_res[1], m_res[1]) && compare_float(s_res[0], m_res[0]))
    {
        printf("\n\nThe solution using both the methods match!");
        plotGraph(a, b, c, p, q, r, s_res[0], s_res[1]);
    }
    else
        printf("\n\nThe solution using both methods are different!");
    return 0;
}
```