ECE 2305

Introduction to C Programming

Programming Project 08

Gaussian Elimination

Program Features: Two dimensional arrays, Dynamic Memory Allocation.

Create a C++ program that will find the solution to a set of linearly independent equations using Gaussian elimination. Write the program in general terms so that it can be applied to any set of equations. The set of n equations have the form

$a0,0\ x0$ + $a0,1\ x1$ + $a0,2\ x2$ + ⋯ + $a0,n-1\ xn-1$ = $b0$

$a1,0\ x0$ + $a1,1\ x1$ + $a1,2\ x2$ + ⋯ + $a1,n-1\ xn-1$ = $b1$

$a2,0\ x0$ + $a2,1\ x1$ + $a2,2\ x2$ + ⋯ + $a2,n-1\ xn-1$ = $b2$

⋮

$an-1,0\ x0$ + $an-1,1\ x1$ + $an-1,2\ x2$ + ⋯ + $an-1,n-1\ xn-1$ = $bn-1$

where aij and bk are coefficients that are input by the user and xk are unknowns for which the application will solve.

If the set of cannot be solved (because of a division by zero error), the application should display a warning message.

A general set of linearly independent equations may be written in the form

[

$a0,0$

(0) $a0,1$

(0) $a0,2$

(0) ⋯ $a0,n-1$

(0)

$a1,0$

(0) $a_{1,1}$

(0) $a_{1,2}$

(0) $\cdots$ $a_{1,n-1}$

(0)

$a_{2,0}$

(0) $a_{2,1}$

(0) $a_{2,2}$

(0) $\cdots$ $a_{2,n-1}$

(0)

$\vdots$ $\vdots$ $\vdots$ $\ddots$ $\vdots$

$a_{n-1,0}$

(0) $a_{n-1,1}$

(0) $a_{n-1,2}$

(0) $\cdots$ $a_{n-1,n-1}$

(0)] [

x_0

(0)

x_1

(0)

x_2

(0)

$\vdots$

x_{n-1}

(0)]

=

[

$b_0^{(0)}$

$b_1^{(0)}$

$b_2^{(0)}$

⋮

$b_{n-1}^{(0)}$]

An algorithm for Gaussian elimination (written in pseudo-code) is as follows:[2]

```
For rows $k = 0$ to $n - 2$, assuming $a_{k,k}^{(k)} \neq 0$
    For rows $i = k + 1$ to $n - 1$, define the row multipliers
        $m = \frac{a_{i,k}^{(k)}}{a_{k,k}^{(k)}}$
        For columns $j = k$ to $n - 1$
            $a_{i,j}^{(k+1)} = a_{i,j}^{(k)} - m a_{k,j}^{(k)}$
        Next $j$
        $b_i^{(k+1)} = b_i^{(k)} - m b_k$
```

```
(k)
Next i
Next k
```

After this procedure the matrix equation takes the form

[

a0,0

(0) a0,1

(0) a0,2

(0) ⋯ a0,n−1

(0)

0 a1,1

(1) a1,2

(1) ⋯ a1,n−1

(1)

0 0 a2,2

(2) ⋯ a2,n−1

(2)

⋮ ⋮ ⋮ ⋱ ⋮

0 0 0 ⋯ an−1,n−1

(n−1)] [

x0

(0)

x1

(0)

x2

(0)

⋮

$xn-1$

(0)]

=

[

$b0$

(0)

$b1$

(1)

$b2$

(2)

⋮

$bn-1$

($n-1$)]

The unknowns xk may be found by first finding the nth unknown

$xn-1 = bn-1$

$an-1,n-1$

The remainder may be found by the formula (again in pseudo-code):

For $k = n - 2$ to 0 step -1

$xk = 1$

ak,k

[$bk - \sum ak,j$

$n-1$

$j=k+1$

xj]

Next k3

To test your program, you may use the set of equations

$-1x0 + 2x1 + 1x2 = 0$

$-2x0 + 2x1 + 3x2 = 3$

$-1x0 - 3x1 + 0x2 = 2$

which have the solution $x0 = 1$, $x1 = -1$, and $x2 = 1$.

Write the program to allow the user to input the number of equations in the set and allow the user to input all the coefficients. Test the program using the set of 3 equations shown above and the sets of equations shown below.

Set of 4 equations:

$-2x0 + 01x1 - 1x2 + 2x3 = 5$

$-4x0 + 05x1 - 3x2 + 6x3 = 9$

$-2x0 + 05x1 - 2x2 + 6x3 = 4$

$-4x0 + 11x1 - 4x2 + 8x3 = 2$

Set of 5 equations

$2x0 - 2x1 + 2x2 - 2x3 + 2x4 = 6$

$2x0 + 0x1 + 0x2 + 0x3 + 0x4 = 2$

$2x0 + 0x1 + 2x2 - 2x3 + 2x4 = 6$

$2x0 + 0x1 + 2x2 + 0x3 + 0x4 = 0$

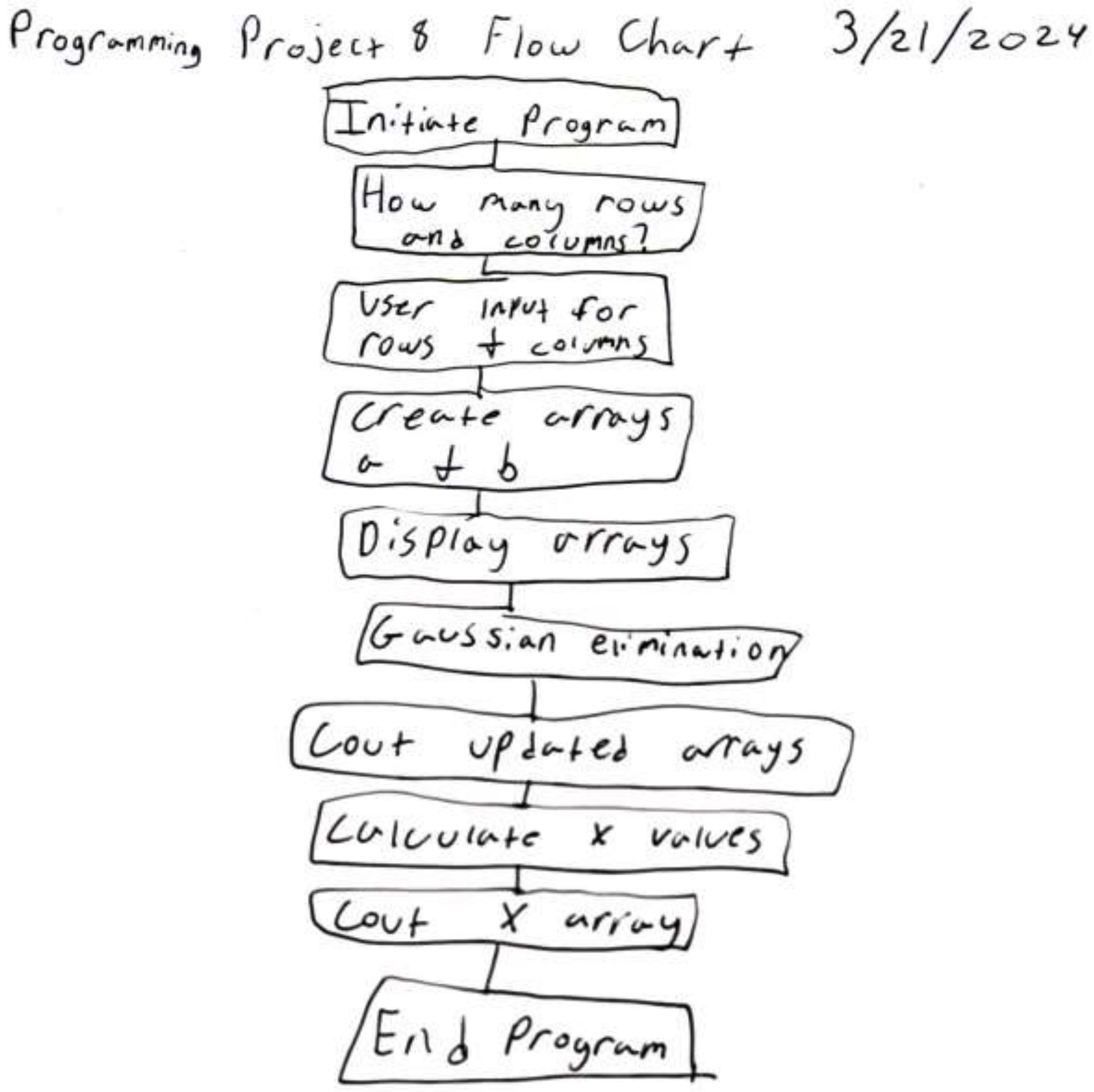
$2x0 + 0x1 + 2x2 + 0x3 + 2x4 = 6$

Document your program with the following:

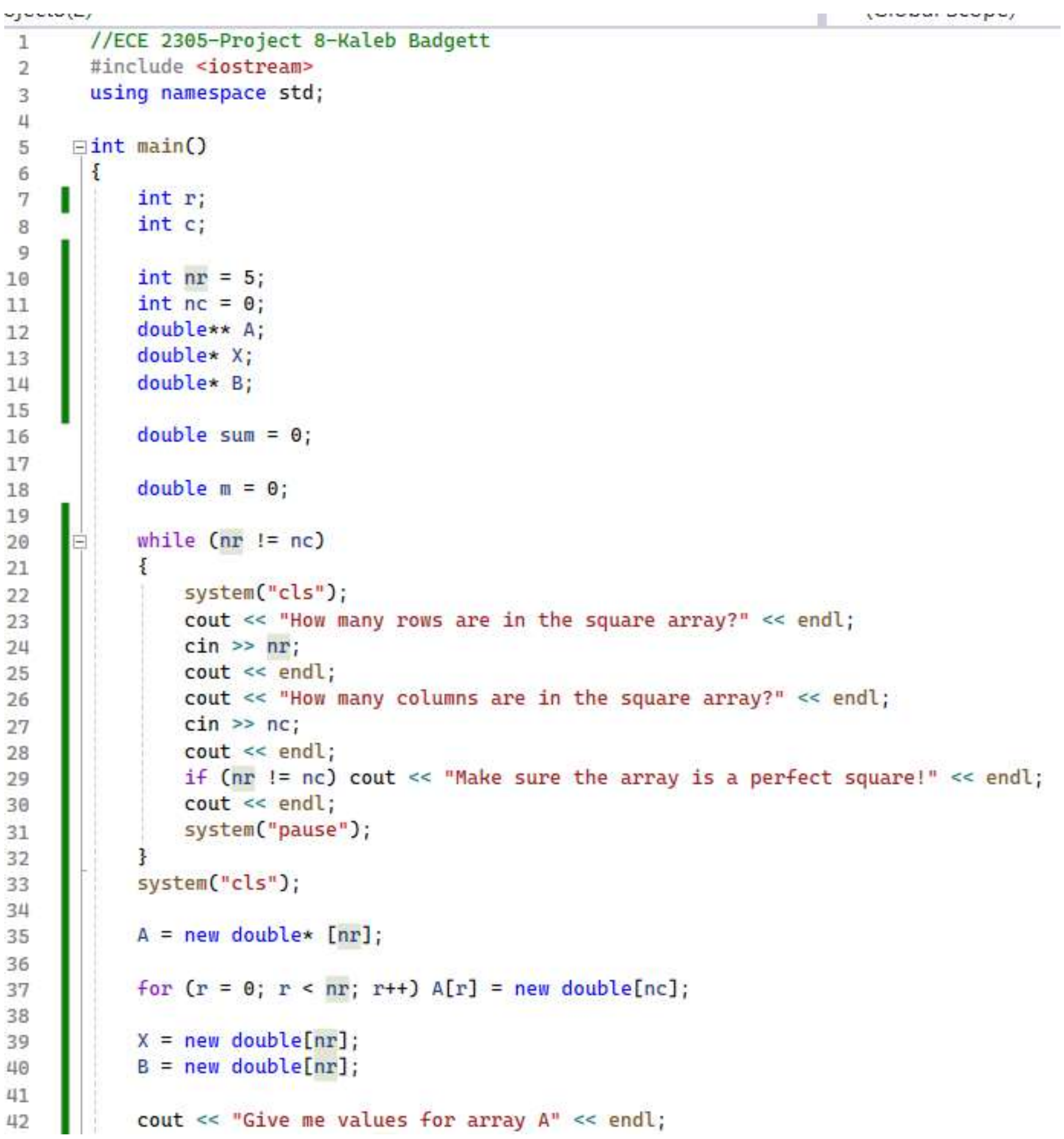
A. A written description of the purpose of the program and the structure of the programming.

The purpose of this program is to solve a gaussian elimination problem of any size. It uses arrays, a while loop, several for loops, and several pointers to solve this problem.

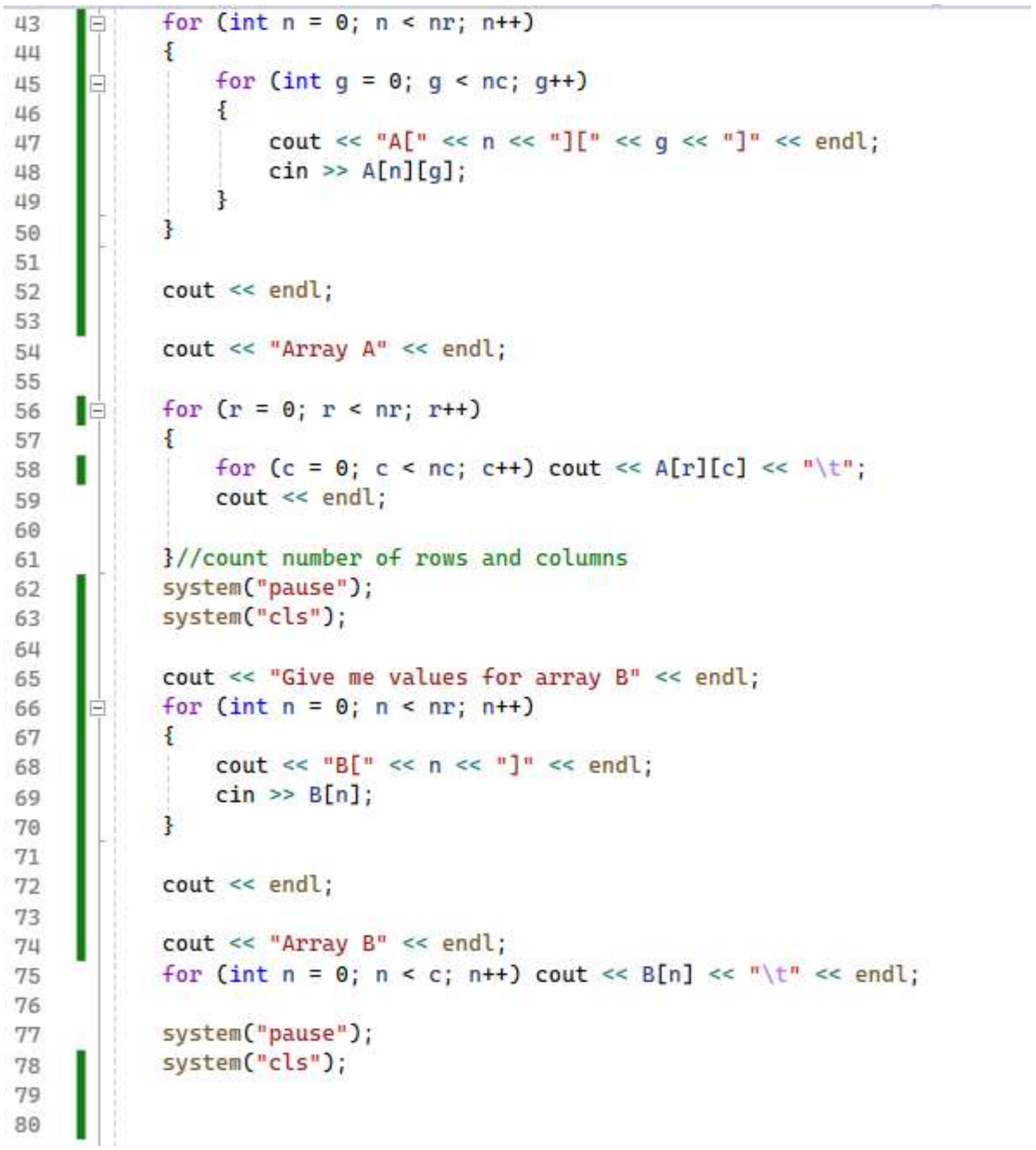
B. A flowchart to graphically display the structure of the program.



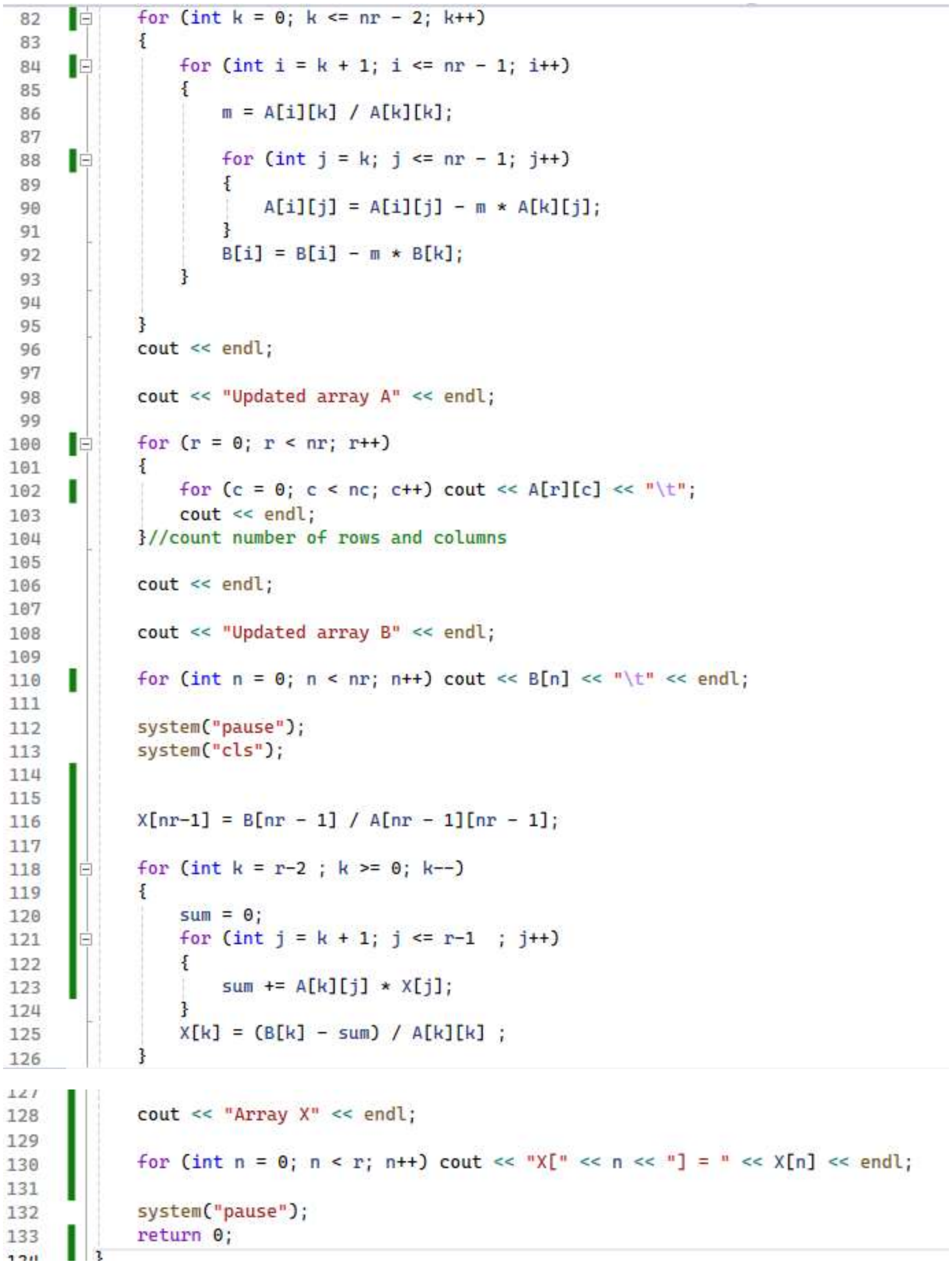
C. The code listing.



```
//ECE 2305-Project 8-Kaleb Badgett
#include <iostream>
using namespace std;

int main()
{
    int r;
    int c;

    int nr = 5;
    int nc = 0;
    double** A;
    double* X;
    double* B;

    double sum = 0;

    double m = 0;

    while (nr != nc)
    {
        system("cls");
        cout << "How many rows are in the square array?" << endl;
        cin >> nr;
        cout << endl;
        cout << "How many columns are in the square array?" << endl;
        cin >> nc;
        cout << endl;
        if (nr != nc) cout << "Make sure the array is a perfect square!" << endl;
        cout << endl;
        system("pause");
    }
    system("cls");

    A = new double* [nr];

    for (r = 0; r < nr; r++) A[r] = new double[nc];

    X = new double[nr];
    B = new double[nr];

    cout << "Give me values for array A" << endl;
```

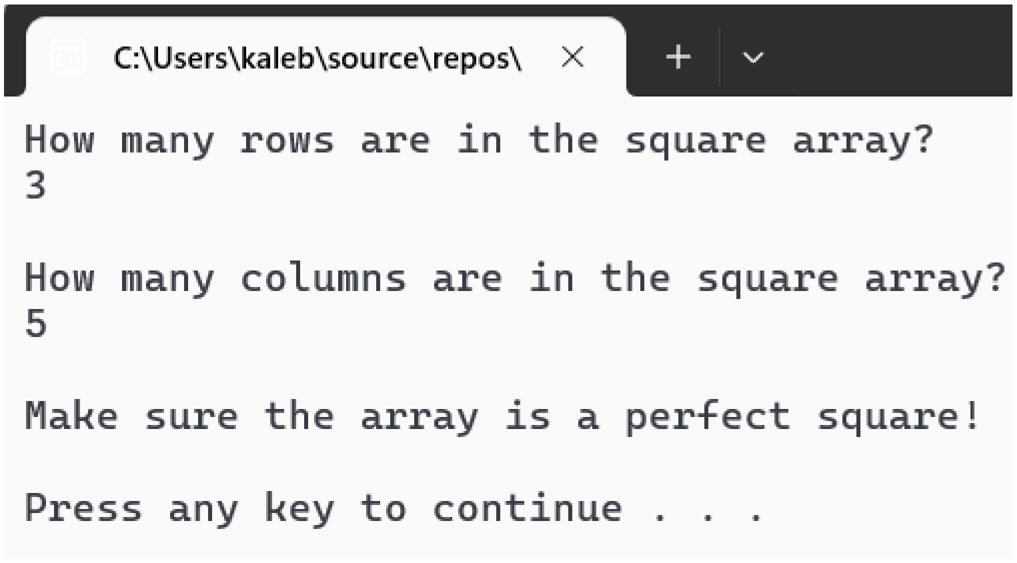


```cpp
    for (int n = 0; n < nr; n++)
    {
        for (int g = 0; g < nc; g++)
        {
            cout << "A[" << n << "][" << g << "]" << endl;
            cin >> A[n][g];
        }
    }

    cout << endl;

    cout << "Array A" << endl;

    for (r = 0; r < nr; r++)
    {
        for (c = 0; c < nc; c++) cout << A[r][c] << "\t";
        cout << endl;

    }//count number of rows and columns
    system("pause");
    system("cls");

    cout << "Give me values for array B" << endl;
    for (int n = 0; n < nr; n++)
    {
        cout << "B[" << n << "]" << endl;
        cin >> B[n];
    }

    cout << endl;

    cout << "Array B" << endl;
    for (int n = 0; n < c; n++) cout << B[n] << "\t" << endl;

    system("pause");
    system("cls");
```

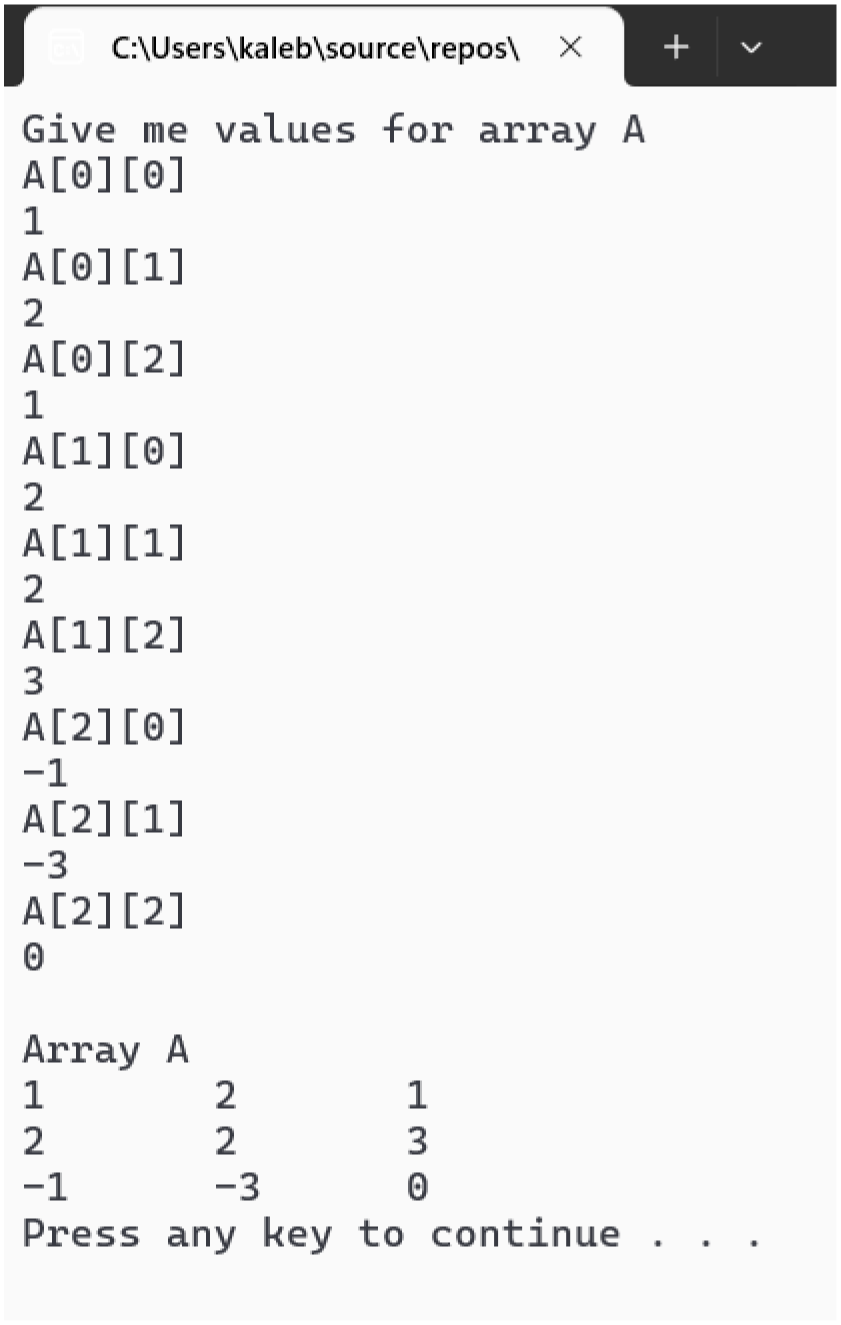


```cpp
    for (int k = 0; k <= nr - 2; k++)
    {
        for (int i = k + 1; i <= nr - 1; i++)
        {
            m = A[i][k] / A[k][k];

            for (int j = k; j <= nr - 1; j++)
            {
                A[i][j] = A[i][j] - m * A[k][j];
            }
            B[i] = B[i] - m * B[k];
        }

    }
    cout << endl;

    cout << "Updated array A" << endl;

    for (r = 0; r < nr; r++)
    {
        for (c = 0; c < nc; c++) cout << A[r][c] << "\t";
        cout << endl;
    }//count number of rows and columns

    cout << endl;

    cout << "Updated array B" << endl;

    for (int n = 0; n < nr; n++) cout << B[n] << "\t" << endl;

    system("pause");
    system("cls");


    X[nr-1] = B[nr - 1] / A[nr - 1][nr - 1];

    for (int k = r-2 ; k >= 0; k--)
    {
        sum = 0;
        for (int j = k + 1; j <= r-1  ; j++)
        {
            sum += A[k][j] * X[j];
        }
        X[k] = (B[k] - sum) / A[k][k] ;
    }

    cout << "Array X" << endl;

    for (int n = 0; n < r; n++) cout << "X[" << n << "] = " << X[n] << endl;

    system("pause");
    return 0;
}
```

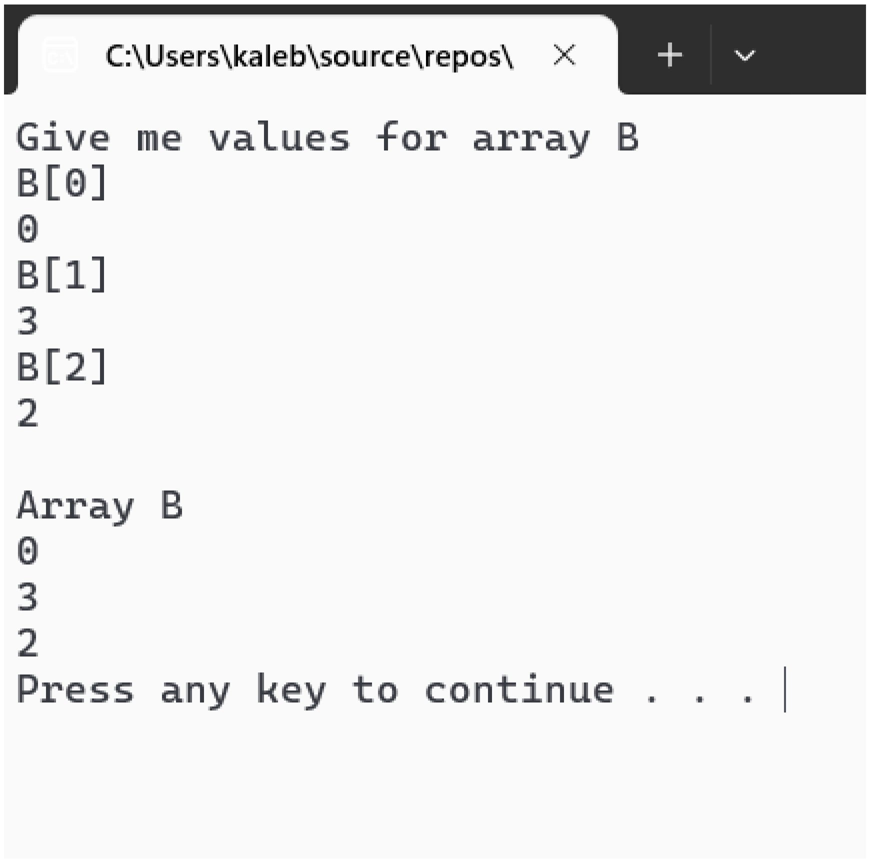
D. Screenshots demonstrating the operation of the program using the sets of 3, 4 and 5 equations that are shown above.



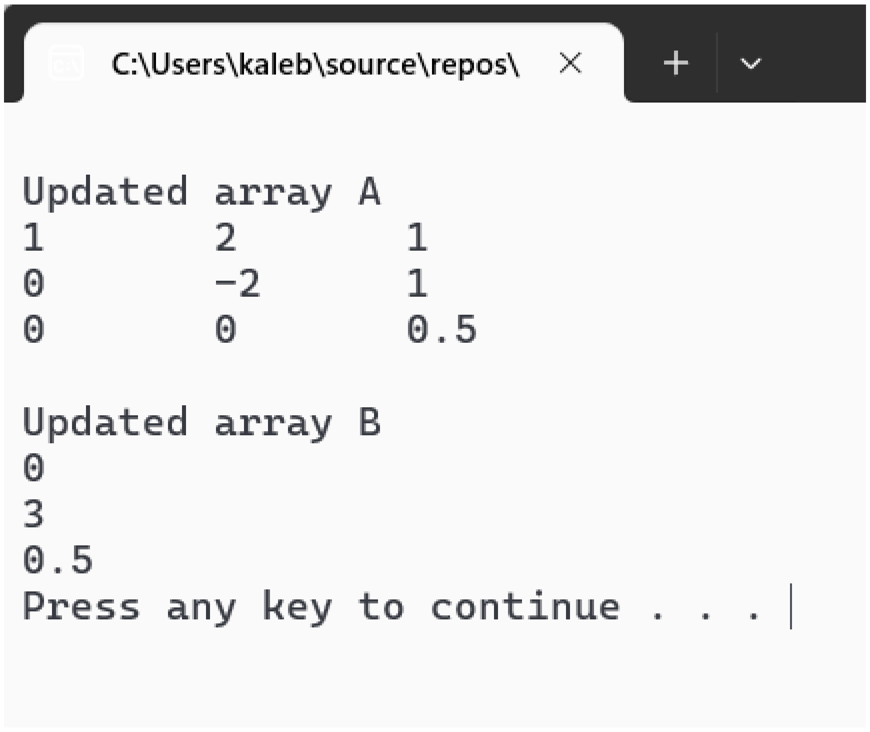
```
How many rows are in the square array?
3

How many columns are in the square array?
5

Make sure the array is a perfect square!

Press any key to continue . . .
```



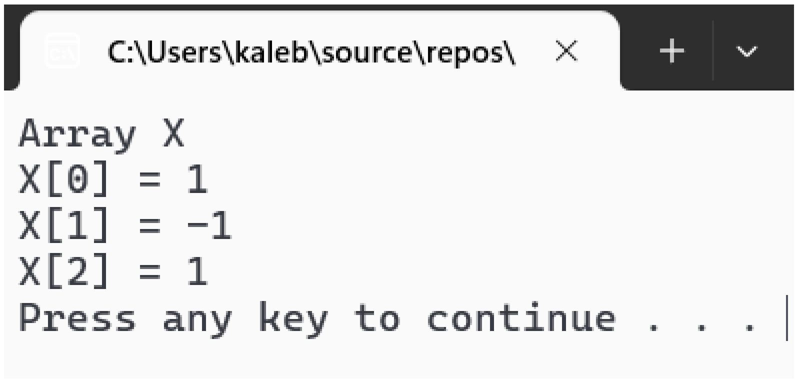
```
Give me values for array A
A[0][0]
1
A[0][1]
2
A[0][2]
1
A[1][0]
2
A[1][1]
2
A[1][2]
3
A[2][0]
-1
A[2][1]
-3
A[2][2]
0

Array A
1       2       1
2       2       3
-1      -3      0
Press any key to continue . . .
```



```
Give me values for array B
B[0]
0
B[1]
3
B[2]
2

Array B
0
3
2
Press any key to continue . . .
```

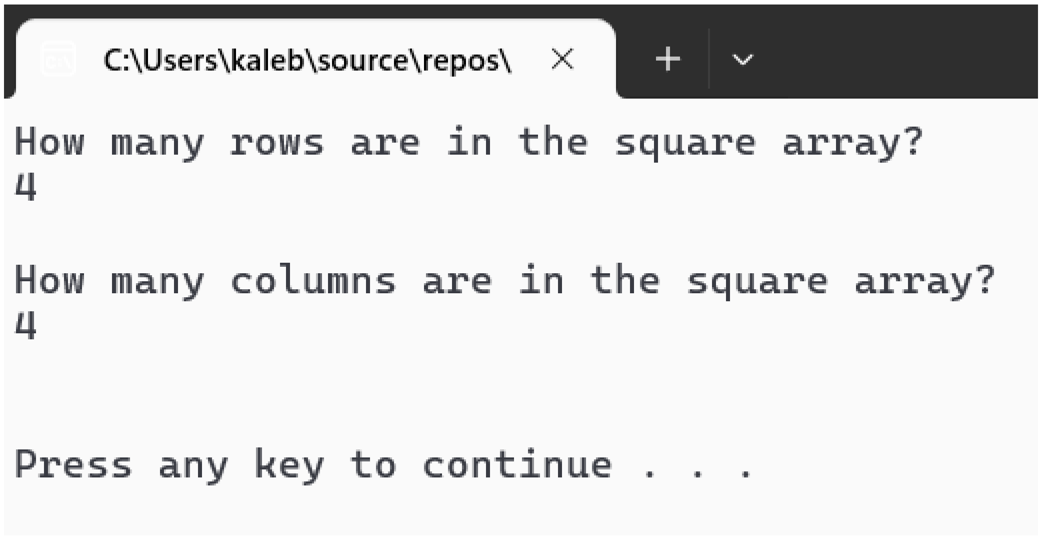


```

Updated array A
1       2       1
0       -2      1
0       0       0.5

Updated array B
0
3
0.5
Press any key to continue . . .
```



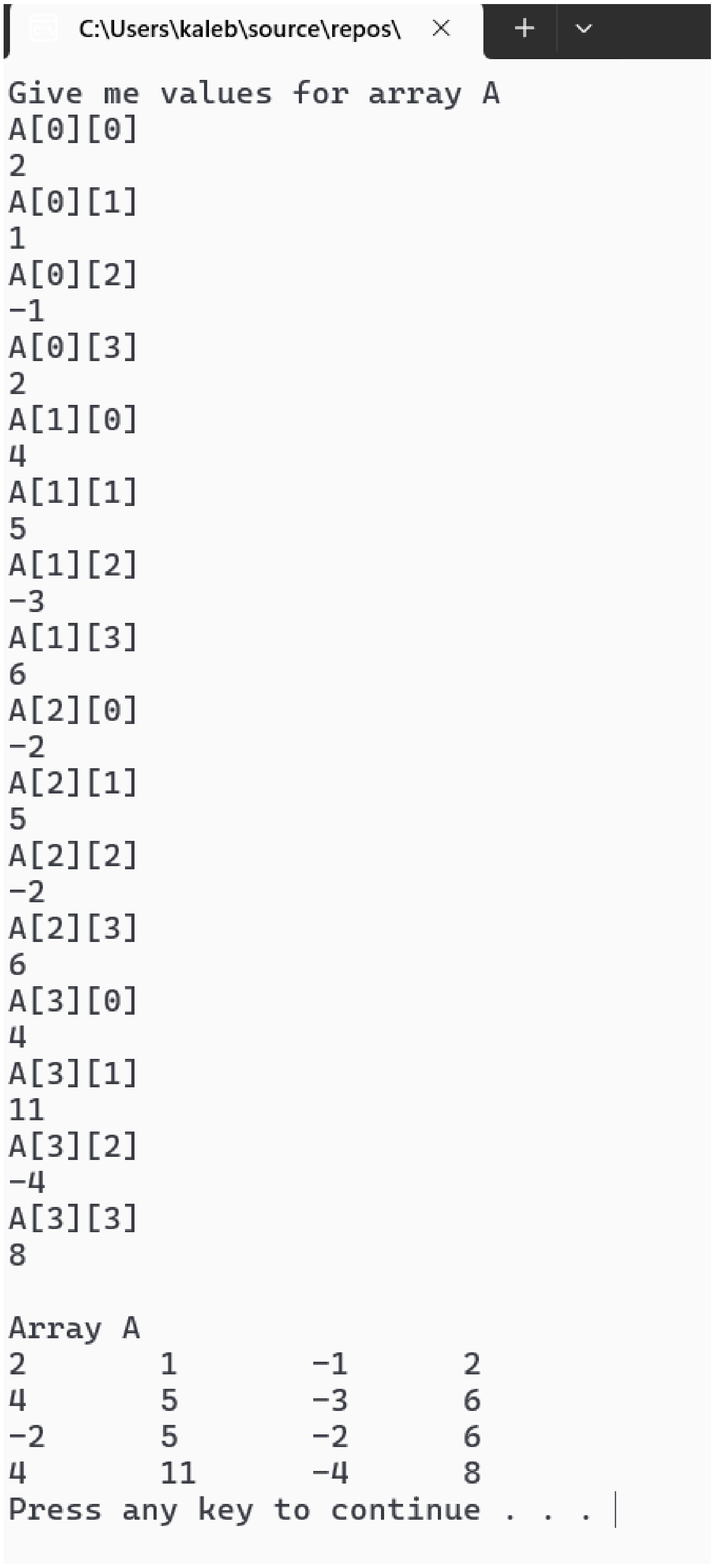
C:\Users\kaleb\source\repos\

```
Array X
X[0] = 1
X[1] = -1
X[2] = 1
Press any key to continue . . .
```

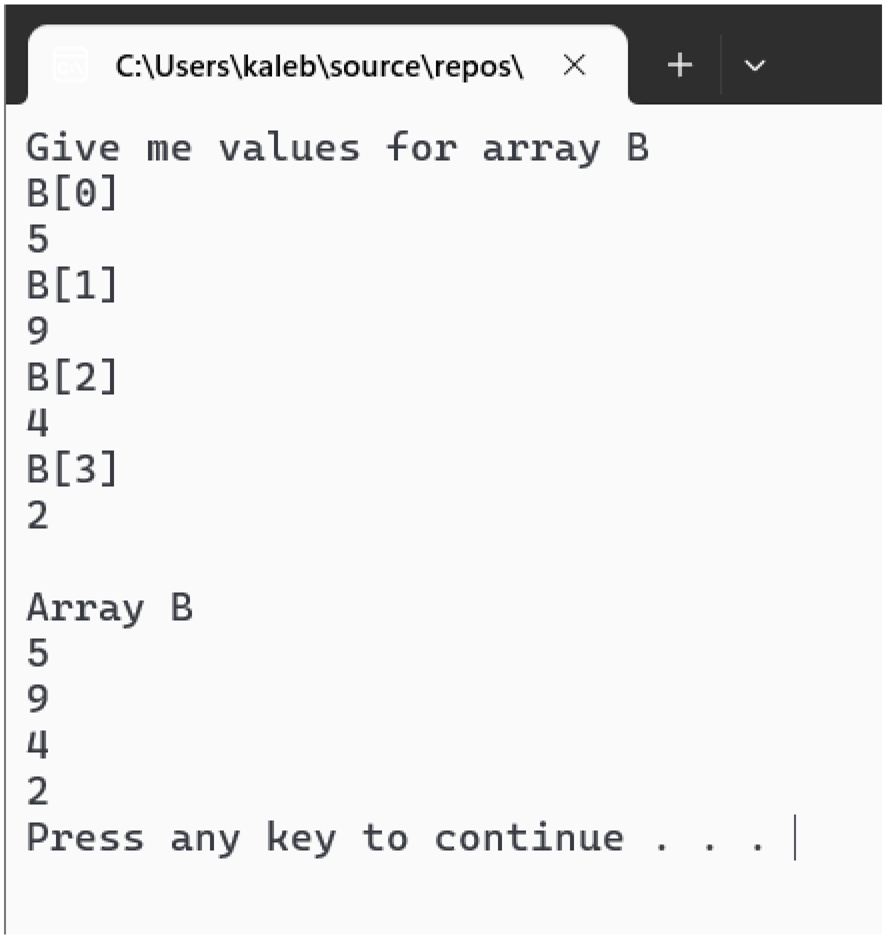


C:\Users\kaleb\source\repos\

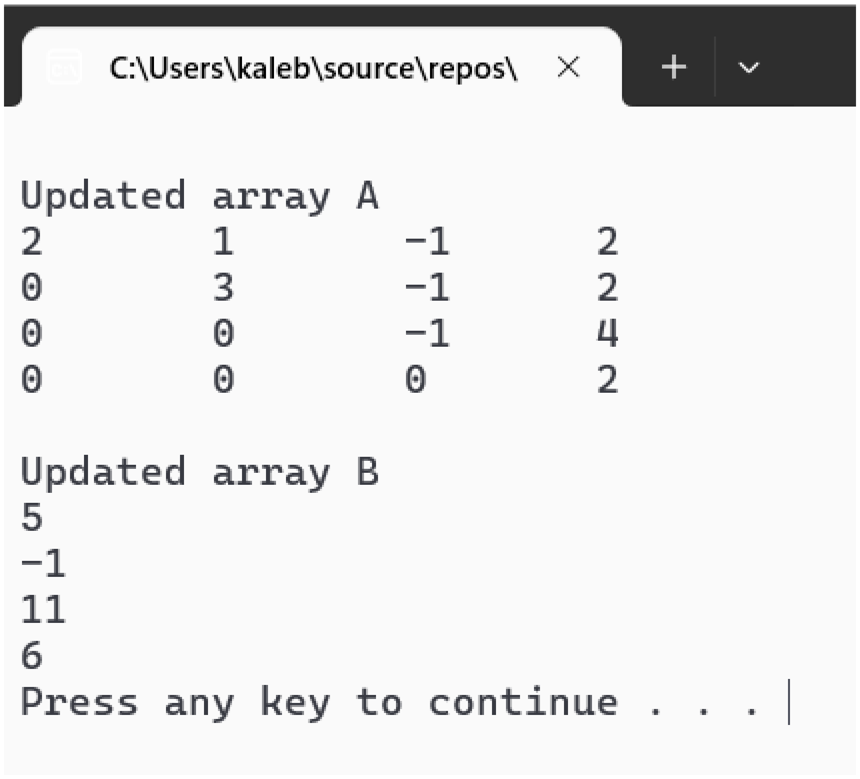
```
How many rows are in the square array?
4

How many columns are in the square array?
4


Press any key to continue . . .
```



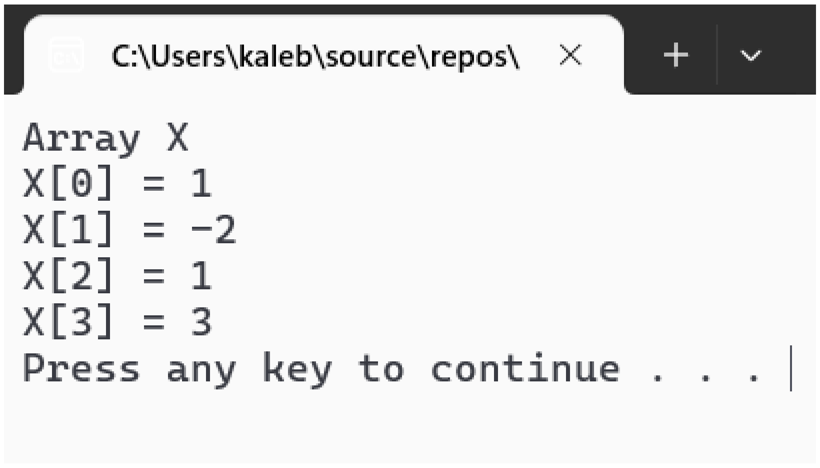
```
Give me values for array A
A[0][0]
2
A[0][1]
1
A[0][2]
-1
A[0][3]
2
A[1][0]
4
A[1][1]
5
A[1][2]
-3
A[1][3]
6
A[2][0]
-2
A[2][1]
5
A[2][2]
-2
A[2][3]
6
A[3][0]
4
A[3][1]
11
A[3][2]
-4
A[3][3]
8

Array A
2       1       -1      2
4       5       -3      6
-2      5       -2      6
4       11      -4      8
Press any key to continue . . .
```



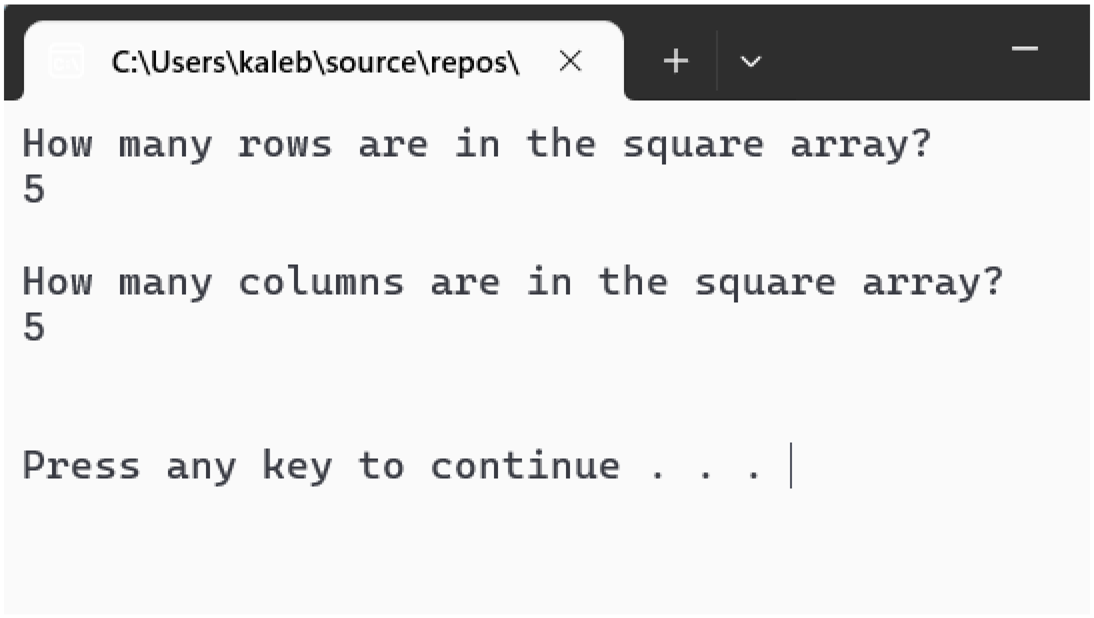
```
Give me values for array B
B[0]
5
B[1]
9
B[2]
4
B[3]
2

Array B
5
9
4
2
Press any key to continue . . .
```



```
Updated array A
2       1       -1      2
0       3       -1      2
0       0       -1      4
0       0       0       2

Updated array B
5
-1
11
6
Press any key to continue . . .
```

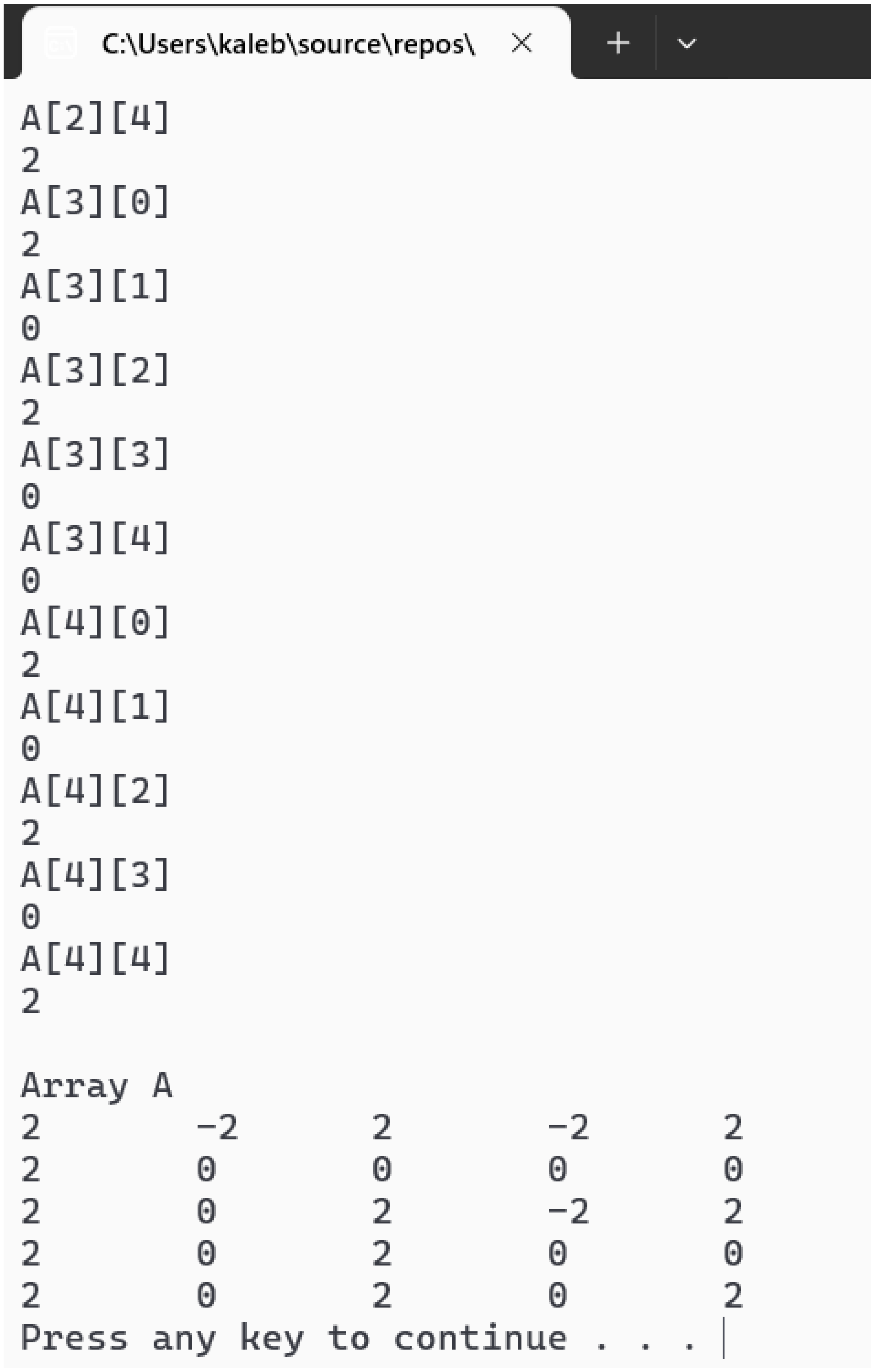


```
Array X
X[0] = 1
X[1] = -2
X[2] = 1
X[3] = 3
Press any key to continue . . .
```

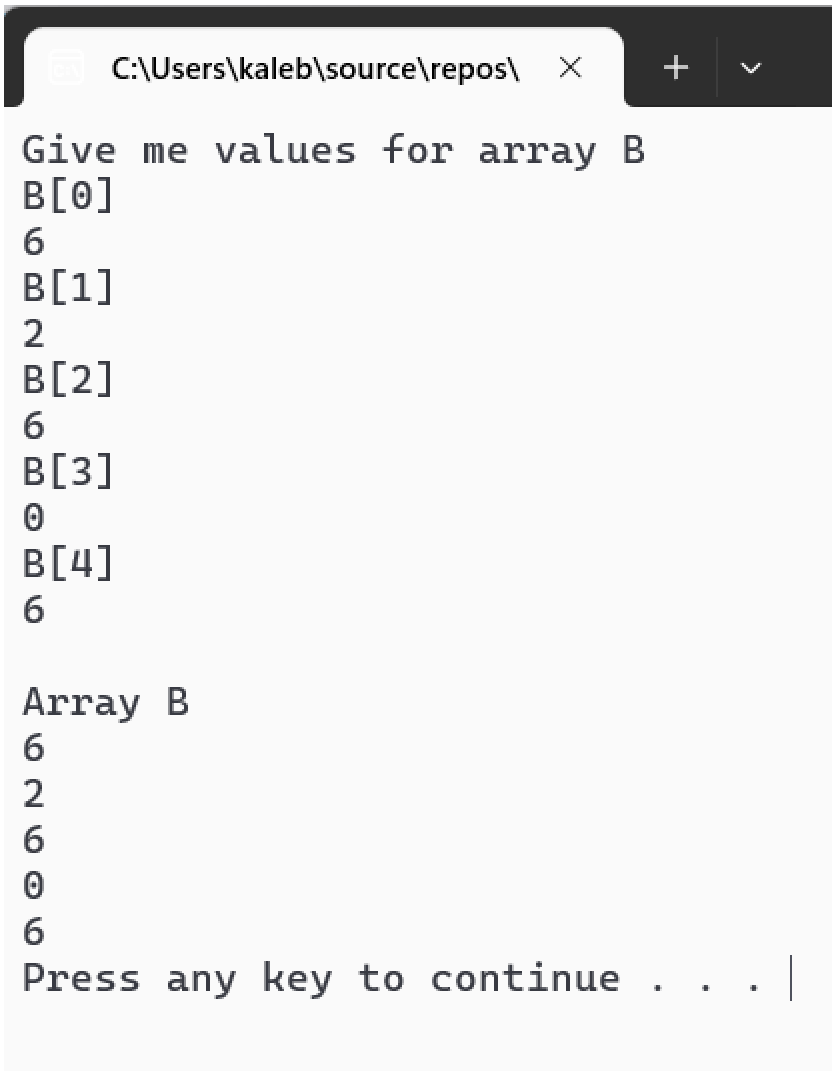


C:\Users\kaleb\source\repos\

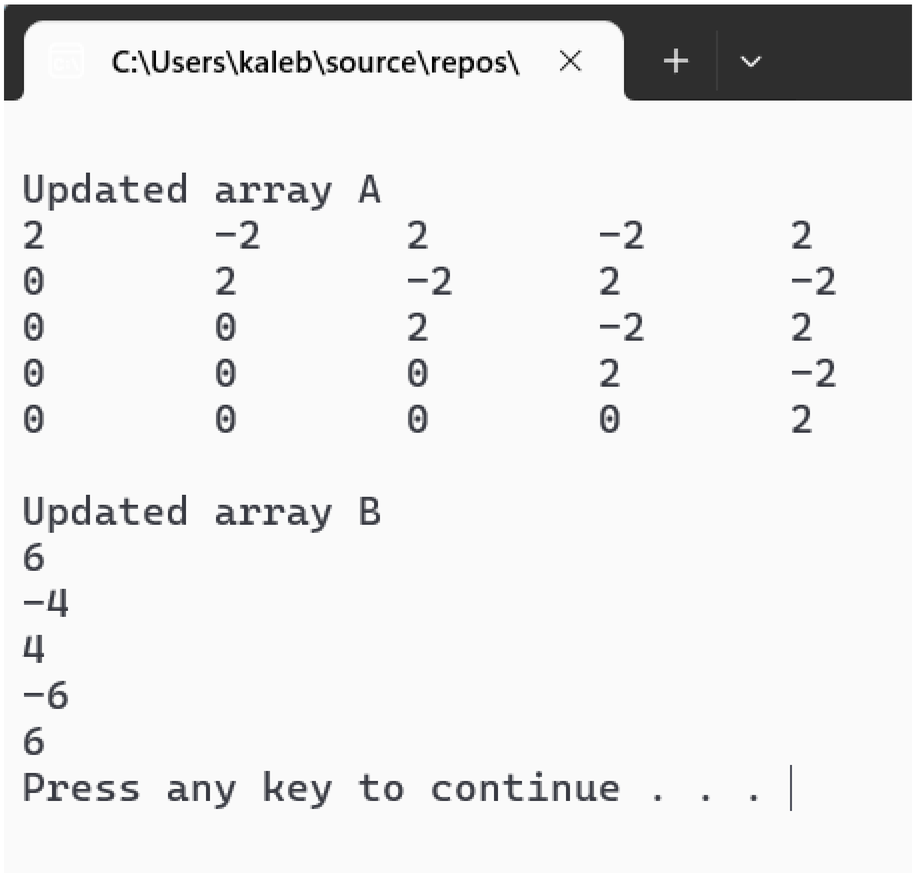
```
How many rows are in the square array?
5

How many columns are in the square array?
5


Press any key to continue . . .
```



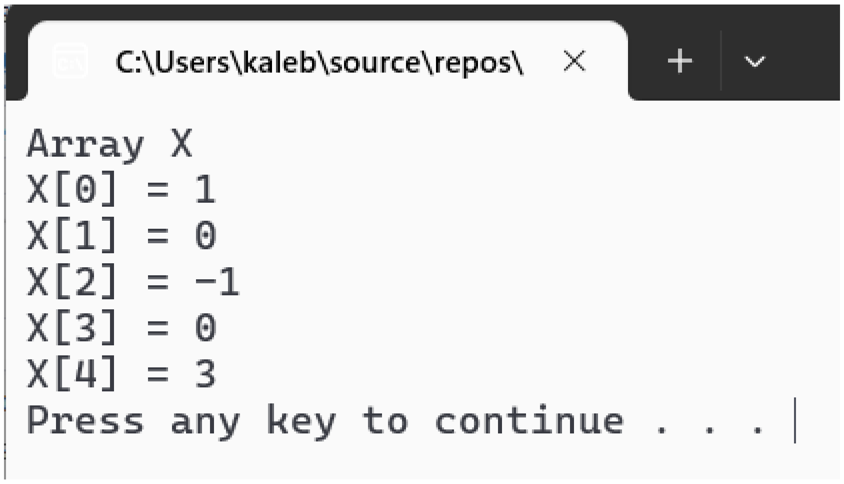
```
A[2][4]
2
A[3][0]
2
A[3][1]
0
A[3][2]
2
A[3][3]
0
A[3][4]
0
A[4][0]
2
A[4][1]
0
A[4][2]
2
A[4][3]
0
A[4][4]
2

Array A
2       -2      2       -2      2
2       0       0       0       0
2       0       2       -2      2
2       0       2       0       0
2       0       2       0       2
Press any key to continue . . .
```



```
Give me values for array B
B[0]
6
B[1]
2
B[2]
6
B[3]
0
B[4]
6

Array B
6
2
6
0
6
Press any key to continue . . .
```



```
Updated array A
2       -2      2       -2      2
0       2       -2      2       -2
0       0       2       -2      2
0       0       0       2       -2
0       0       0       0       2

Updated array B
6
-4
4
-6
6
Press any key to continue . . .
```



```
Array X
X[0] = 1
X[1] = 0
X[2] = -1
X[3] = 0
X[4] = 3
Press any key to continue . . .
```