Question 3.8:

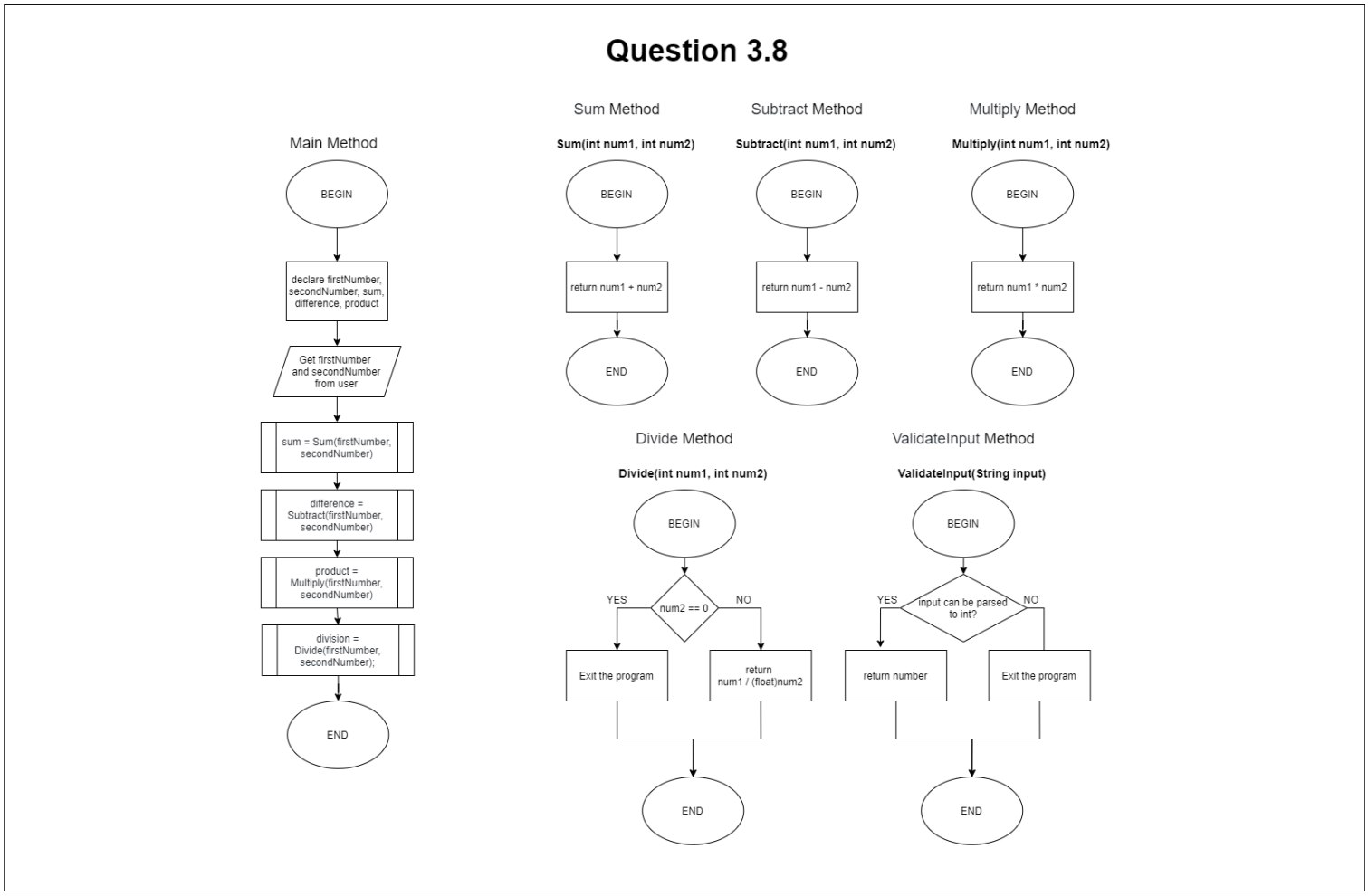
Write an application that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference and quotient of the two numbers.

Solution 3.8:

PSEUDOCODE:

- Two inputs should be gotten from UI and should be validated that inputs are integer
- Sum <- firstNumber + SecondNumber
- Difference <- firstNumber - SecondNumber
- Product <- firstNumber * SecondNumber
- Division <- if secondNumber == 0 ? Throw new Exception(DivideByZero) : firstNumber / (float)SecondNumber

FLOWCHART:



CODE:

```
HW_3.8 | HW_3_8.Program | Main(string[] args)
```

```
using System;
/*
 *
 * 3.8 Write an application that asks the user to enter two numbers, obtains the two numbers from
 * the user and prints the sum, product, difference and quotient of the two numbers.
 *
 */
namespace HW_3._8
{
    class Program
    {
        // main entry point for the application
        static void Main(string[] args)
        {
            int firstNumber, secondNumber, sum, difference, product;
            float division;

            String description = "Application that obtains the two numbers from the user and prints the sum, product, difference and quotient of the two numbers: \n";
            Console.WriteLine(description.ToUpperInvariant());

            // requesting two numbers from user to calculate summation, subtraction, difference and division
            // also inputs will be checked with ValidateInput() method to ensure that inputs are numbers
            Console.WriteLine("Please enter the first number:");
            firstNumber = ValidateInput(Console.ReadLine());

            Console.WriteLine("Please enter the second number:");
            secondNumber = ValidateInput(Console.ReadLine());

            sum = Sum(firstNumber, secondNumber);
            difference = Subtract(firstNumber, secondNumber);
            product = Multiply(firstNumber, secondNumber);
            division = Divide(firstNumber, secondNumber);

            Console.WriteLine("Sum \t\t: " + sum + "\nDifference \t: " + difference + "\nProduct \t: " + product +
                              "\nDivision \t: " + division);

            Console.WriteLine("Press any key to close application");
            Console.ReadKey();
        }

        /// <summary>
        ///     This method takes two integer parameters and returns the sum of them.
        /// </summary>
        /// <param name="num1">First integer number to sum</param>
        /// <param name="num2">Second integer number to sum</param>
        /// <returns>The summation of two doubles</returns>
        static int Sum(int num1, int num2)
        {
            return num1 + num2;
        }

        /// <summary>
        ///     This method takes two integer parameters and returns the subtraction of them.
        /// </summary>
        /// <param name="num1">First integer number to subtract</param>
        /// <param name="num2">Second integer number to subtract</param>
        /// <returns>The subtraction of two doubles</returns>
        static int Subtract(int num1, int num2)
        {
            return num1 - num2;
        }

        /// <summary>
        ///     This method takes two integer parameters and returns the multiplication of them.
        /// </summary>
        /// <param name="num1">First integer number to multiply</param>
        /// <param name="num2">Second integer number to multiply</param>
        /// <returns>The multiplication of two doubles</returns>
        static int Multiply(int num1, int num2)
        {
            return num1 * num2;
        }

        /// <summary>
        ///     This method takes two integer parameters and returns the division of them.
        /// </summary>
        /// <param name="num1">First integer number which is dividend</param>
        /// <param name="num2">Second integer number which is divisor</param>
        /// <returns>The division of two integer number in float type</returns>
        static float Divide(int num1, int num2)
        {
            // divisor can not be zero!
            if (num2 == 0)
            {
                Console.Write("Divisor is 0 and any number can not be divided by 0!! \nPress any key to Exit the program!");
                Console.ReadKey();
                Environment.Exit(0);
                return 0;
            }
            else
            {
                return (num1 / (float)num2);
            }

        }


        /// <summary>
        ///     This method takes an string parameter and check if the content is integer or not
        /// </summary>
        /// <param name="input">User input</param>
```

```
        /// <returns></returns>
        static int ValidateInput(String input)
        {
            int number;
            // if input is parseable, return as integer
            if (Int32.TryParse(input, out number))
            {
                return number;
            }
            else
            {
                Console.Write("Input is not integer!! \nPress any key to Exit the program!");
                Console.ReadKey();
                Environment.Exit(0);
                return 0;
            }
        }
    }
}
```

Outputs of program:

```
C:\Users\YARGICI\Desktop\Visual\Midterm\HW_3.8\bin\Debug\HW_3.8.exe
APPLICATION THAT OBTAINS THE TWO NUMBERS FROM THE USER AND PRINTS THE SUM, PRODUCT, DIFFERENCE AND QUOTIENT OF THE TWO NUMBERS:

Please enter the first number:
1
Please enter the second number:
5
Sum             : 6
Difference      : -4
Product         : 5
Division        : 0,2
Press any key to close application
```

If user enters invalid inputs:

```
C:\Users\YARGICI\Desktop\Visual\Midterm\HW_3.8\bin\Debug\HW_3.8.exe
APPLICATION THAT OBTAINS THE TWO NUMBERS FROM THE USER AND PRINTS THE SUM, PRODUCT, DIFFERENCE AND QUOTIENT OF THE TWO NUMBERS:

Please enter the first number:
a
Input is not integer!!
Press any key to Exit the program!
```

If user enters 0 as divisor:

```
C:\Users\YARGICI\Desktop\Visual\Midterm\HW_3.8\bin\Debug\HW_3.8.exe
APPLICATION THAT OBTAINS THE TWO NUMBERS FROM THE USER AND PRINTS THE SUM, PRODUCT, DIFFERENCE AND QUOTIENT OF THE TWO NUMBERS:

Please enter the first number:
9
Please enter the second number:
0
Divisor is 0 and any number can not be divided by 0!!
Press any key to Exit the program!
```

Question 5.6:

(Pythagorean Triples) A right triangle can have sides that are all integers. A set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of the two sides is equal to the square of the hypotenuse. Write a program to find all Pythagorean triples for side1, side2 and hypotenuse, none larger than 30. Use a triple-nested for loop that tries all possibilities. This is an example of “brute force” computing. You will learn in more advanced computer science courses that there are several problems for which there is no other known algorithmic approach.

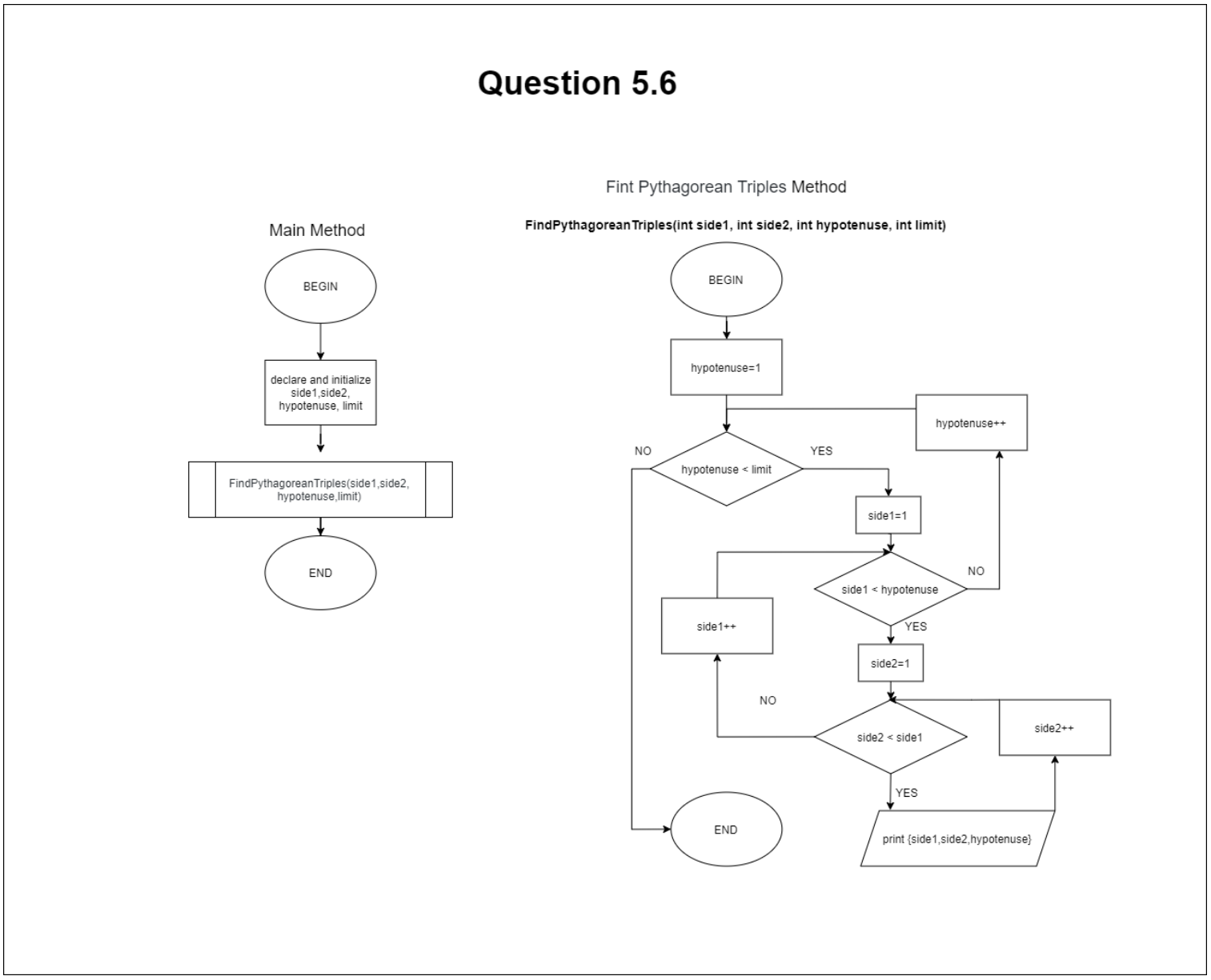
Solution 5.6:

PSEUDOCODE:

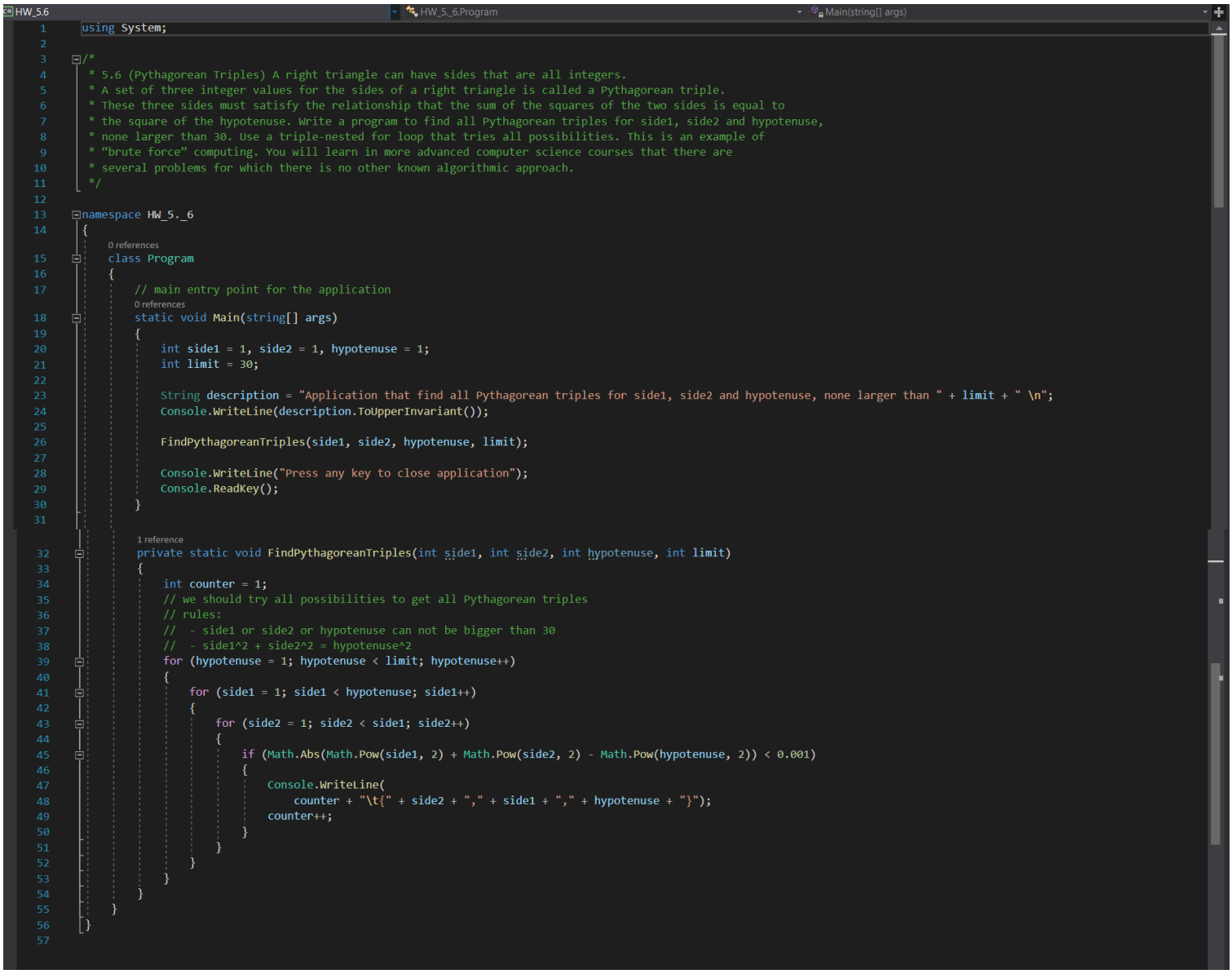
-Pythagorean triples rules:
-- side1 or side2 or hypotenuse can not be bigger than upper limit
-- side1^2 + side2^2 = hypotenuse^2 (from Pythagorean theorem)

- We should indicate upper limit of a side can get
- We should try all possibilities to get all Pythagorean triples in a given range. That can be provided with nested "for"s as author want us. In the last "for" body, we can check Pythagorean theorem and decide if sides provide the rules.

FLOWCHART:

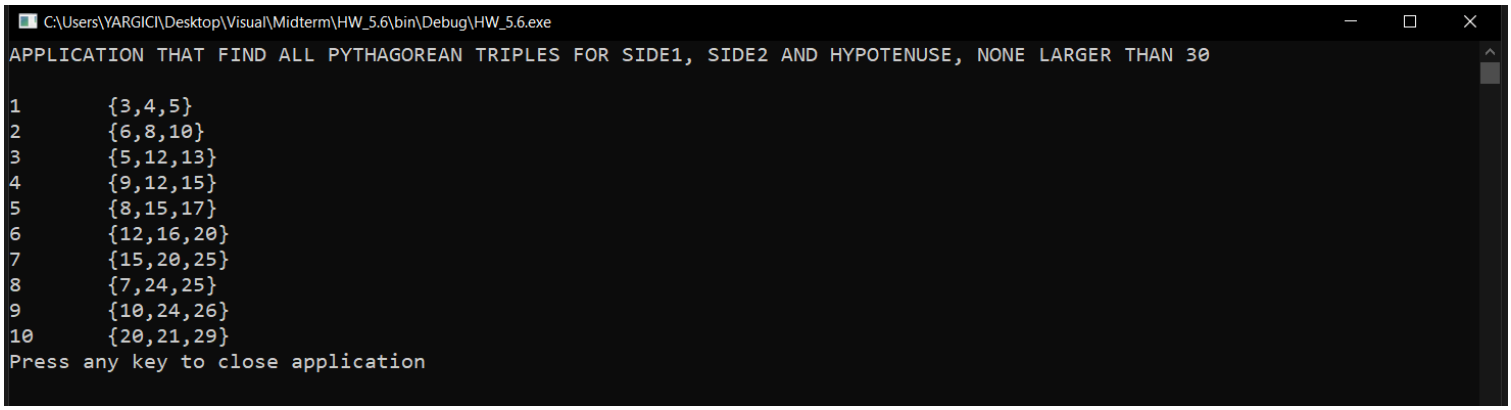


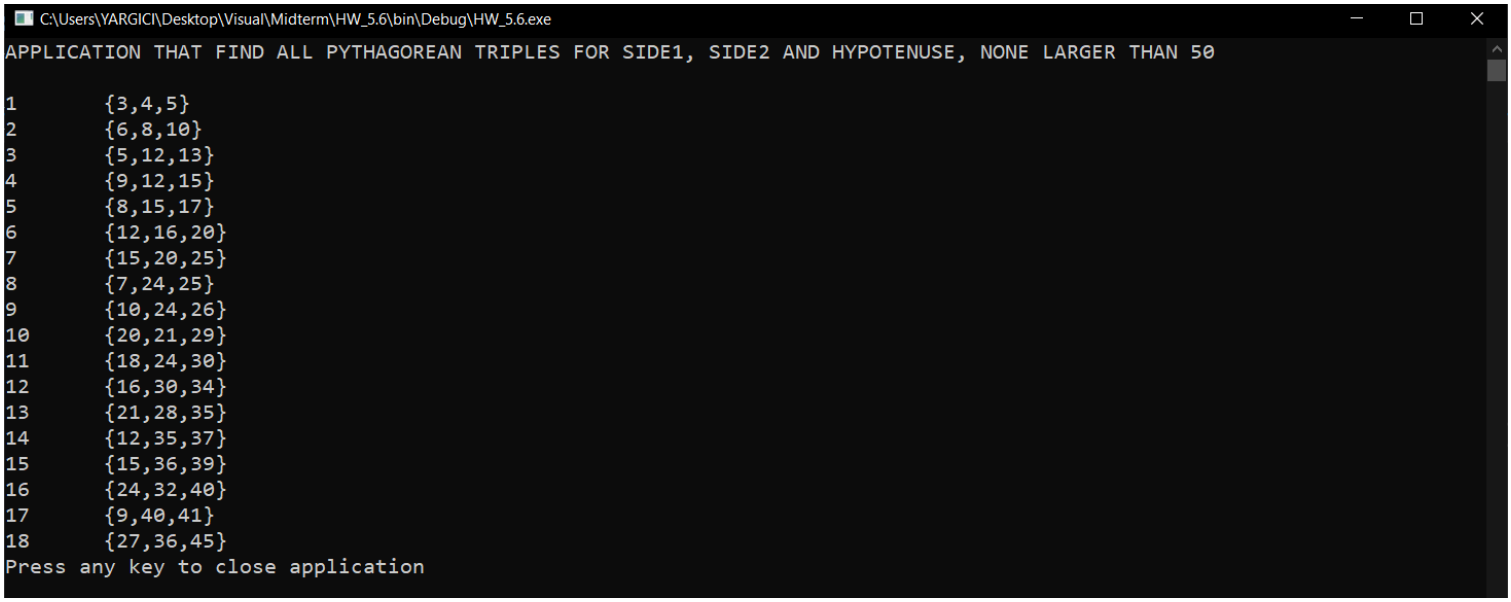
CODE:



```
HW_5.6 | HW_5_6.Program | Main(string[] args)

using System;

/*
 * 5.6 (Pythagorean Triples) A right triangle can have sides that are all integers.
 * A set of three integer values for the sides of a right triangle is called a Pythagorean triple.
 * These three sides must satisfy the relationship that the sum of the squares of the two sides is equal to
 * the square of the hypotenuse. Write a program to find all Pythagorean triples for side1, side2 and hypotenuse,
 * none larger than 30. Use a triple-nested for loop that tries all possibilities. This is an example of
 * "brute force" computing. You will learn in more advanced computer science courses that there are
 * several problems for which there is no other known algorithmic approach.
 */

namespace HW_5._6
{
    0 references
    class Program
    {
        // main entry point for the application
        0 references
        static void Main(string[] args)
        {
            int side1 = 1, side2 = 1, hypotenuse = 1;
            int limit = 30;

            String description = "Application that find all Pythagorean triples for side1, side2 and hypotenuse, none larger than " + limit + " \n";
            Console.WriteLine(description.ToUpperInvariant());

            FindPythagoreanTriples(side1, side2, hypotenuse, limit);

            Console.WriteLine("Press any key to close application");
            Console.ReadKey();
        }

        1 reference
        private static void FindPythagoreanTriples(int side1, int side2, int hypotenuse, int limit)
        {
            int counter = 1;
            // we should try all possibilities to get all Pythagorean triples
            // rules:
            //  - side1 or side2 or hypotenuse can not be bigger than 30
            //  - side1^2 + side2^2 = hypotenuse^2
            for (hypotenuse = 1; hypotenuse < limit; hypotenuse++)
            {
                for (side1 = 1; side1 < hypotenuse; side1++)
                {
                    for (side2 = 1; side2 < side1; side2++)
                    {
                        if (Math.Abs(Math.Pow(side1, 2) + Math.Pow(side2, 2) - Math.Pow(hypotenuse, 2)) < 0.001)
                        {
                            Console.WriteLine(
                                counter + "\t{" + side2 + "," + side1 + "," + hypotenuse + "}");
                            counter++;
                        }
                    }
                }
            }
        }
    }
}
```

Outputs of program:





Question 6.13:

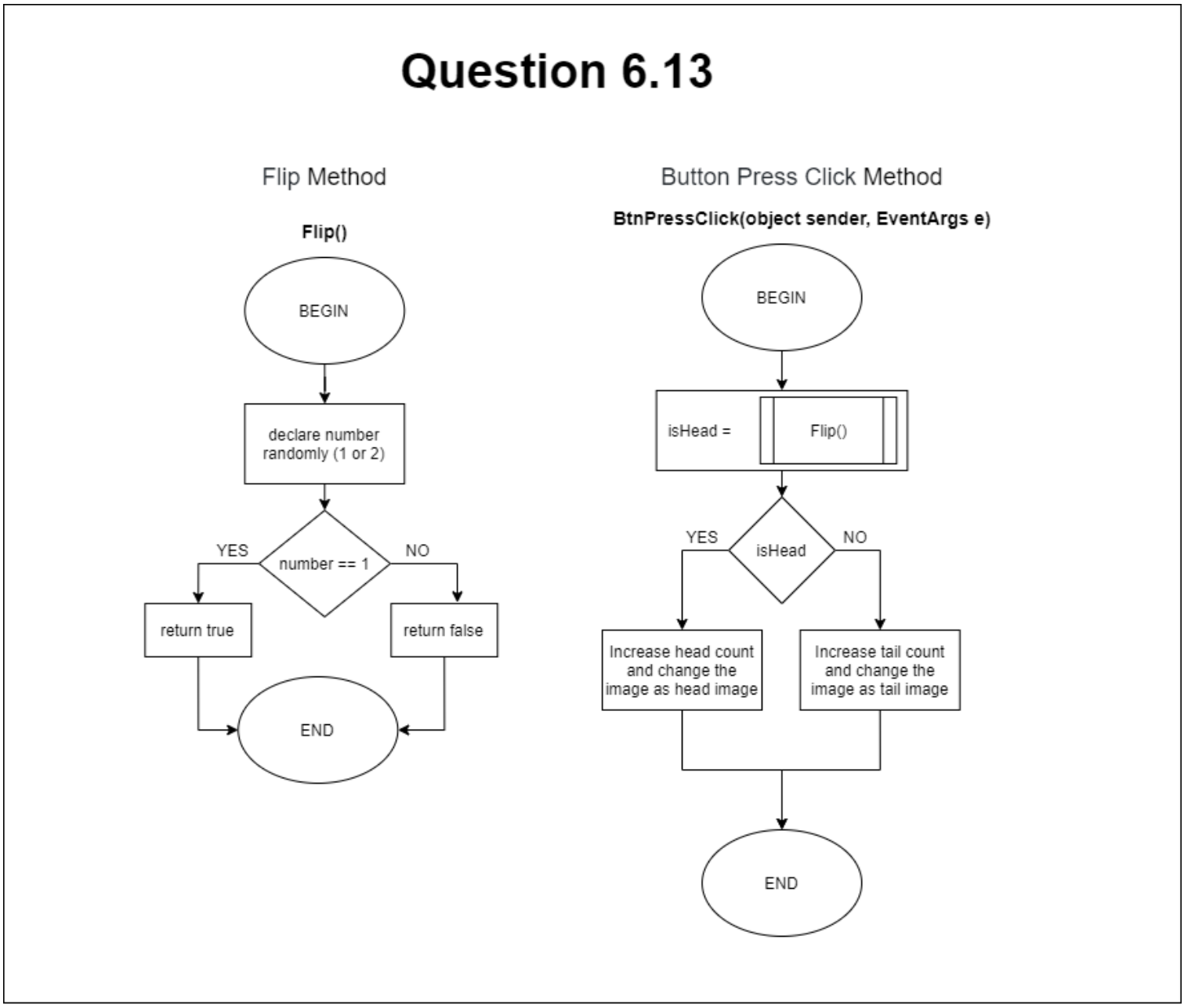
Write an application that simulates coin tossing. Let the program toss the coin each time the user presses the “Toss” button. Count the number of times each side of the coin appears. Display the results. The program should call a separate method Flip that takes no arguments and returns false for tails and true for heads. [Note: If the program realistically simulates the coin tossing, each side of the coin should appear approximately half of the time.]

Solution 6.13:

PSEUDOCODE:

- There should be two images which are head and tail of coin.
- There should be random instance in Flip() method and it should generate two different values (1 and 2), assuming 1 is head and 2 is tail. Flip will return true if result is 1(head).
- If user clicks toss button, pictureBox content should be set according to result of Flip() method.
- According to result of Flip() method, head count or tail count should be increased by 1.

FLOWCHART:



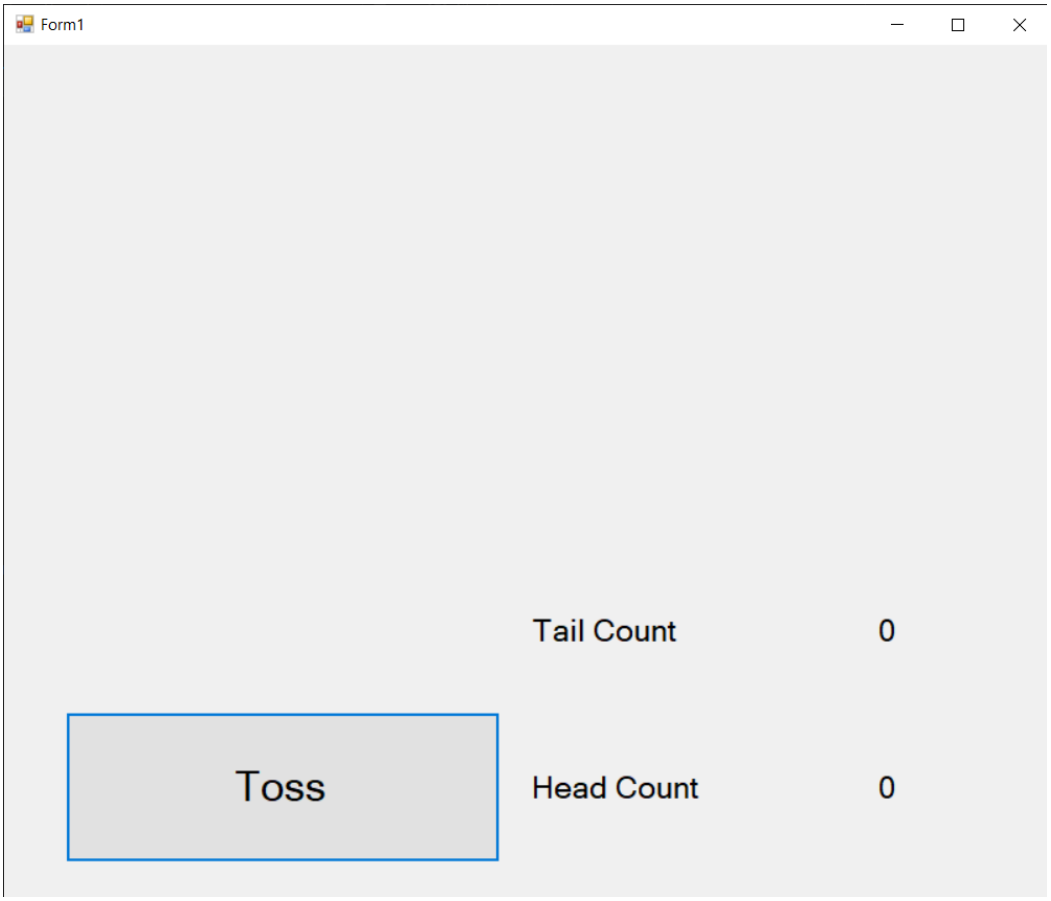
CODE:

```
HW_6.13 | HW_6_13.Form1 | txt
```

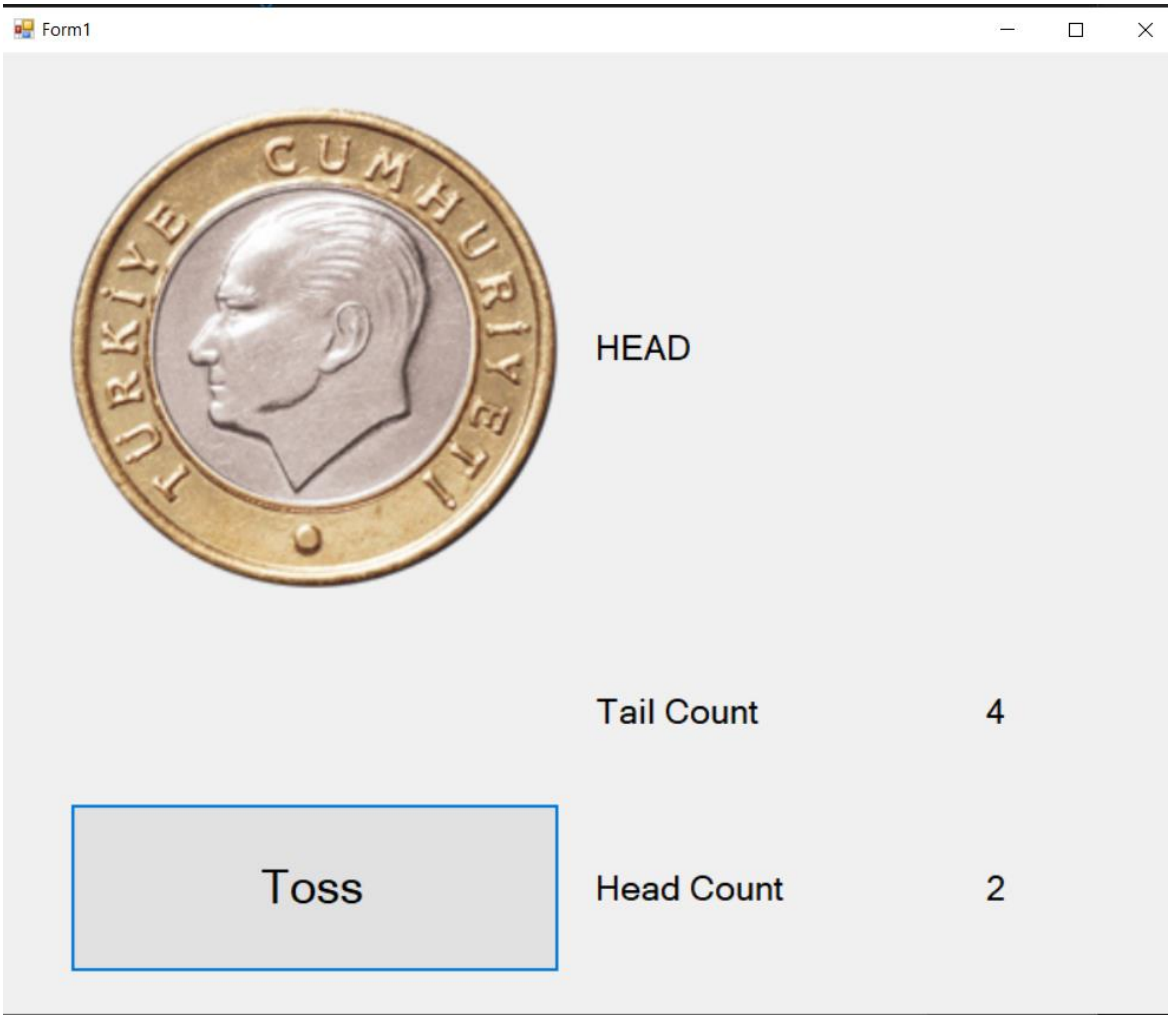
```
using System;
using System.Drawing;
using System.Windows.Forms;

/**
 *
 * 6.13 Write an application that simulates coin tossing. Let the program toss the coin each time the
 * user presses the “Toss” button. Count the number of times each side of the coin appears. Display the
 * results. The program should call a separate method Flip that takes no arguments and returns false
 * for tails and true for heads. [Note: If the program realistically simulates the coin tossing, each side
 * of the coin should appear approximately half of the time.]
 *
 */

namespace HW_6._13
{
    3 references
    public partial class Form1 : Form
    {
        Random random = new Random(); // random instance to generate random number
        string txt = "";

        /// <summary>
        ///     Form initializer
        /// </summary>
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        /// <summary>
        ///     This method generates random number 1 or 2.
        ///     1 means head, 2 means tail.
        /// </summary>
        /// <returns>true if result is head</returns>
        1 reference
        private bool Flip()
        {
            int number = random.Next(1, 3); // returns either 1 or 2
            //head
            if (number == 1) return true;
            //tail
            return false;
        }

        /// <summary>
        ///     This method is listener of btnPress button. Calls Flip() method and sets other tools properties.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 reference
        private void BtnPressClick(object sender, EventArgs e)
        {

            // Flip() returns true if side of coin is head
            bool isHead = Flip();

            if (isHead)
            {
                lblHeadCount.Text = (int.Parse(lblHeadCount.Text) + 1).ToString(); // increasing count
                txt = "HEAD"; // will be used for changing the text of lblText
                pictureBox.Image = new Bitmap(Properties.Resources.head); // changing the image of pictureBox
            }
            else
            {
                lblTailCount.Text = (int.Parse(lblTailCount.Text) + 1).ToString();
                txt = "TAIL";
                pictureBox.Image = new Bitmap(Properties.Resources.tail);

            }
            lblText.Text = txt;
        }

    }
}
```

Outputs of program:

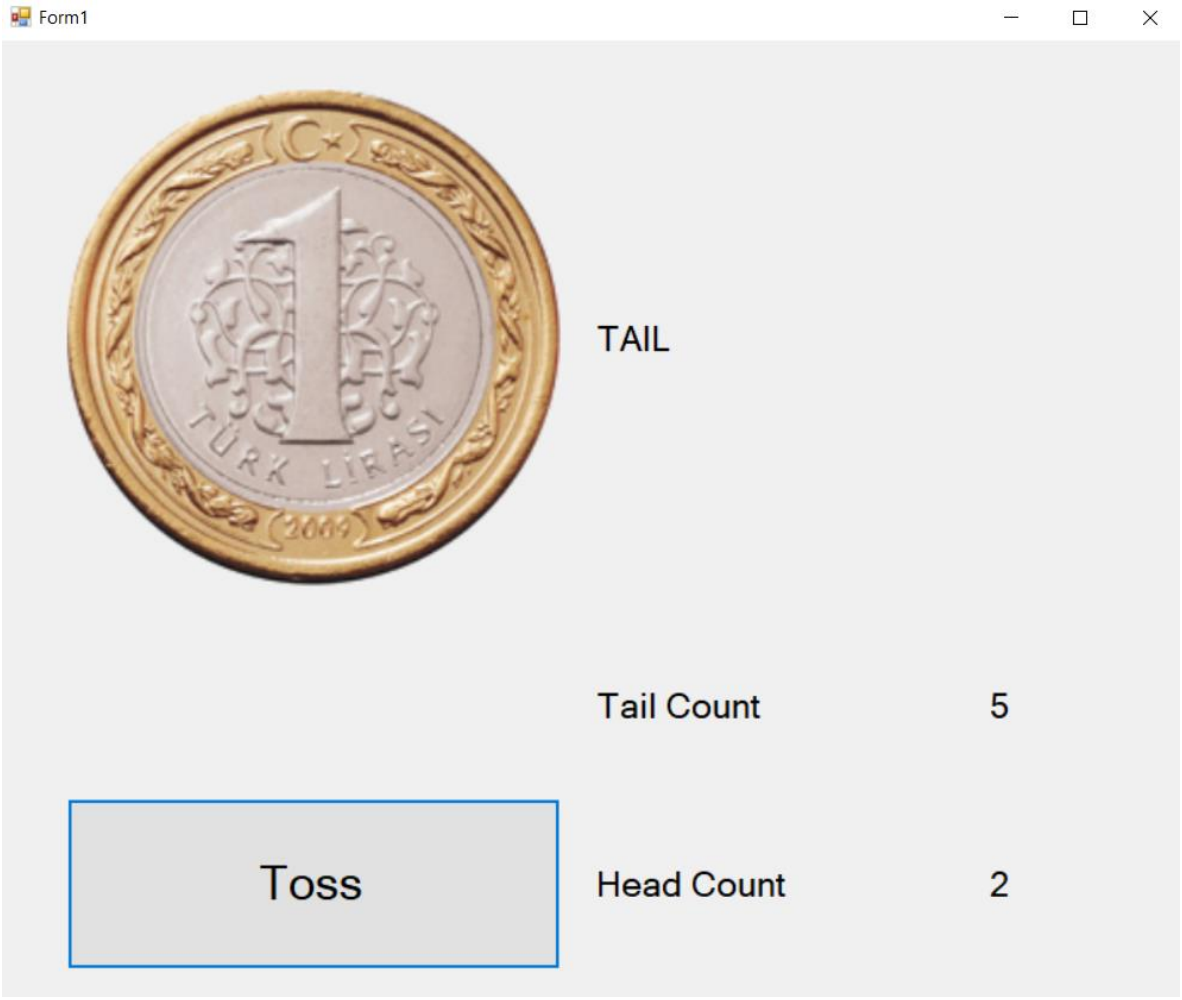
- Starting the application:



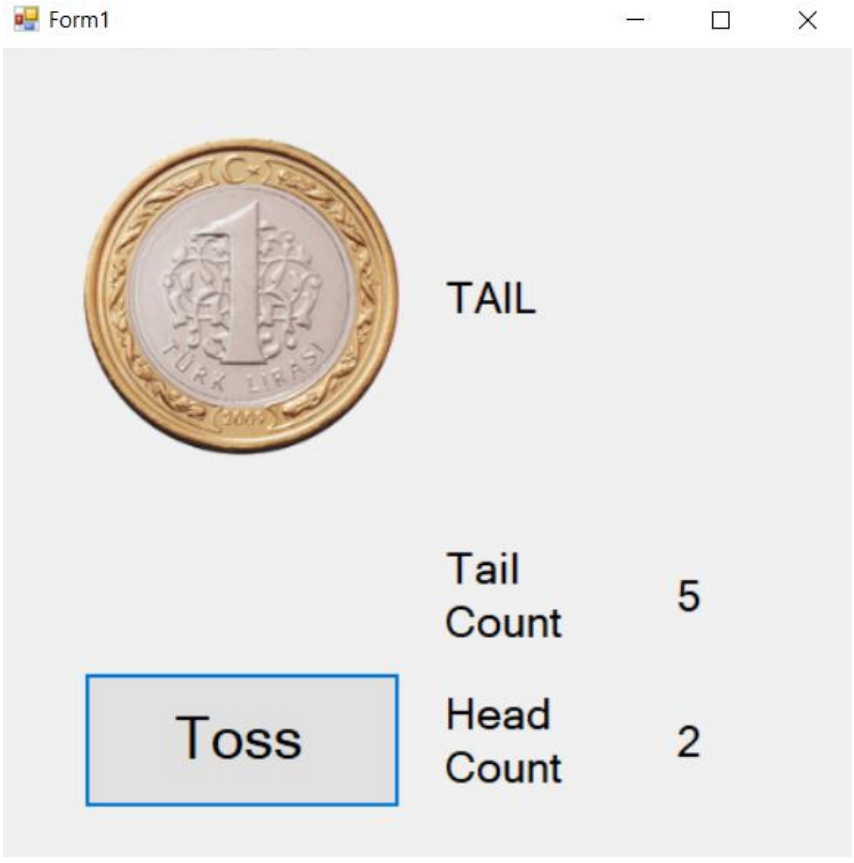
- After toss button click:



- Toss again:



- Responsive design:



Form1

TAIL

Tail Count 5

Toss

Head Count 2

Question 6.16:

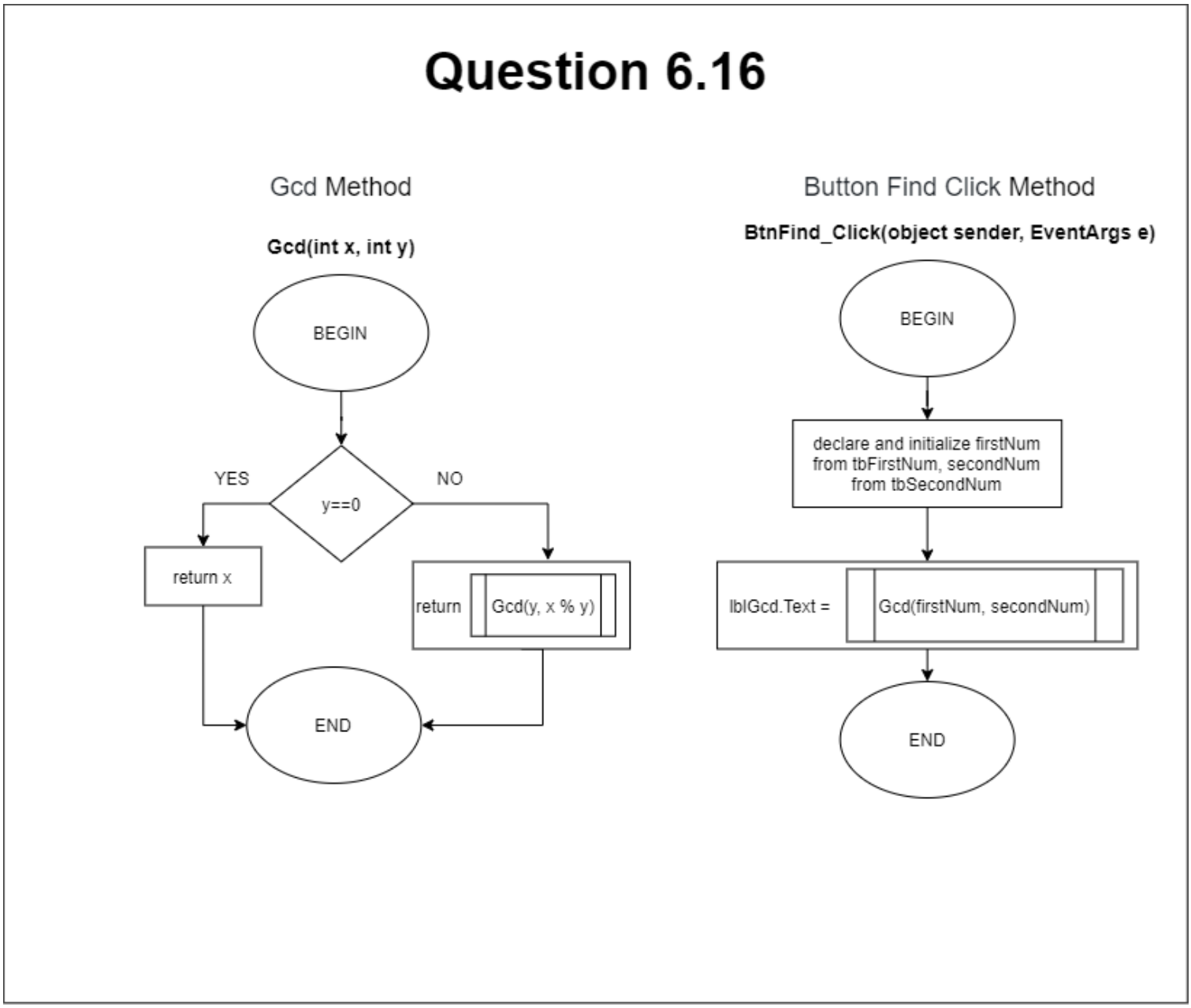
The greatest common divisor of integers x and y is the largest integer that evenly divides both x and y. Write a recursive method Gcd that returns the greatest common divisor of x and y. The Gcd of x and y is defined recursively as follows: If y is equal to 0, then Gcd(x, y) is x; otherwise, Gcd(x, y) is Gcd(y, x % y), where % is the modulus operator.

Solution 6.16:

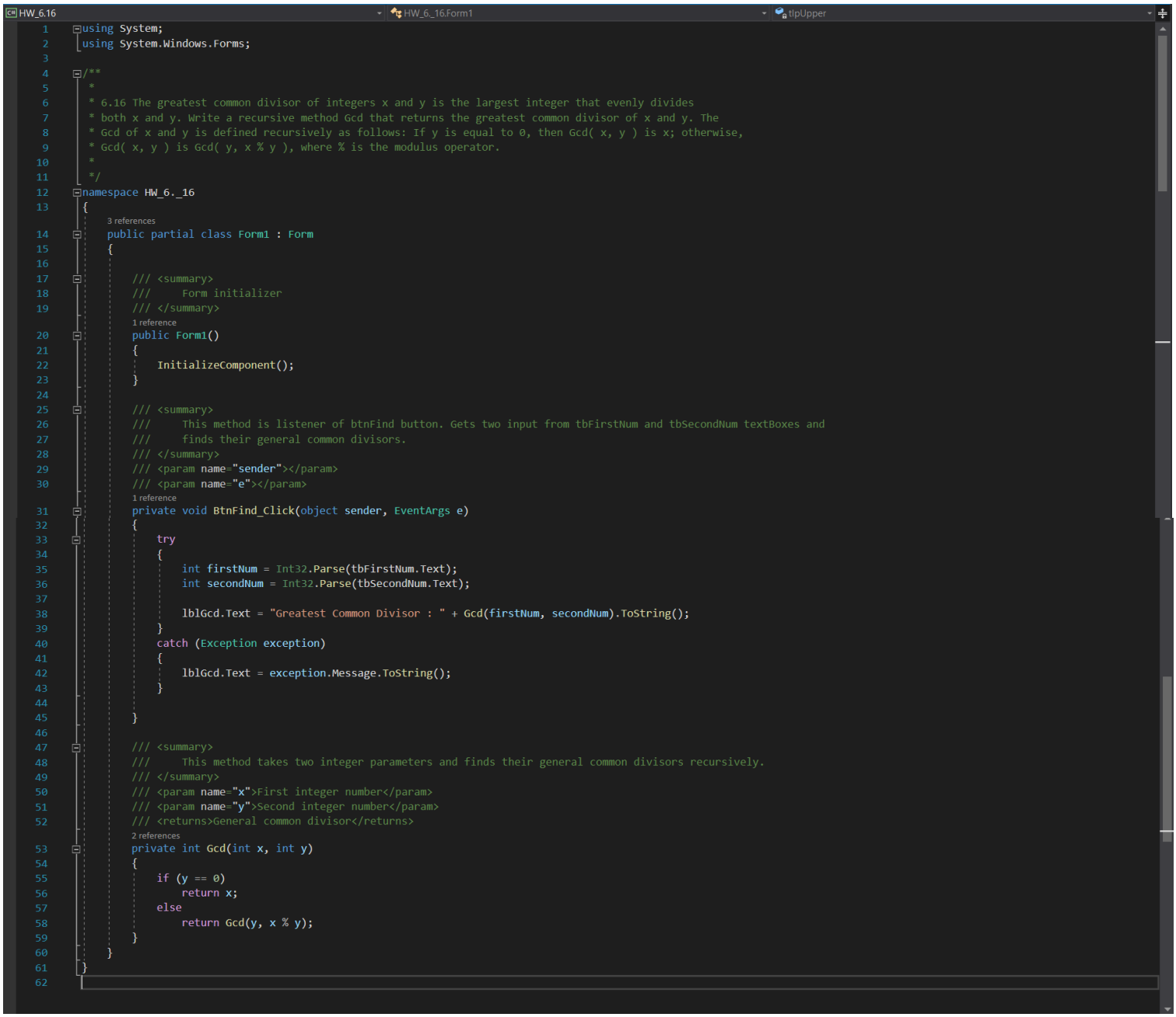
PSEUDOCODE:

- There should be two textBox that user can interact with application and should validated that input is integer.
- Gcd(int x, int y) method should take those two inputs as parameter and find the greatest common divisor recursively. if y==0 ? return x : return Gcd(y,x%y)

FLOWCHART:



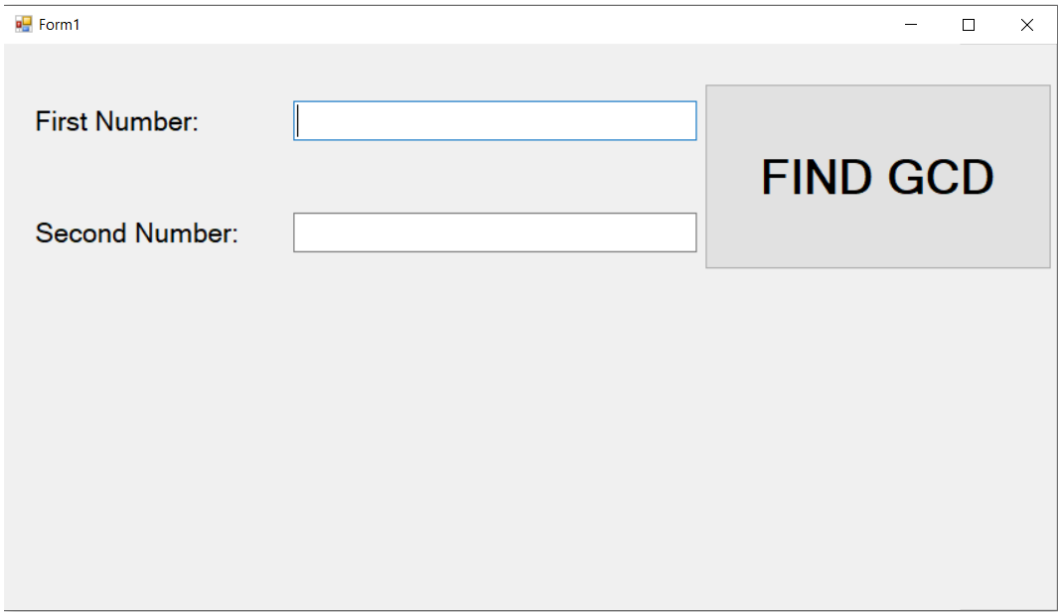
CODE:



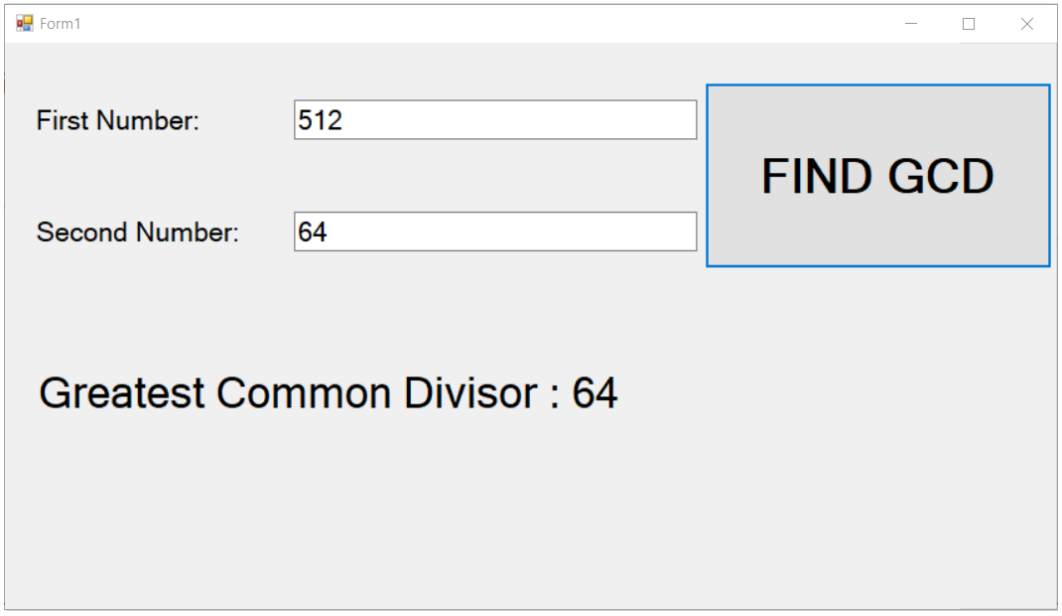
```
HW_6.16 | HW_6_16.Form1 | tlpUpper

using System;
using System.Windows.Forms;

/**
 *
 * 6.16 The greatest common divisor of integers x and y is the largest integer that evenly divides
 * both x and y. Write a recursive method Gcd that returns the greatest common divisor of x and y. The
 * Gcd of x and y is defined recursively as follows: If y is equal to 0, then Gcd( x, y ) is x; otherwise,
 * Gcd( x, y ) is Gcd( y, x % y ), where % is the modulus operator.
 *
 */
namespace HW_6._16
{
    3 references
    public partial class Form1 : Form
    {

        /// <summary>
        ///     Form initializer
        /// </summary>
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        /// <summary>
        ///     This method is listener of btnFind button. Gets two input from tbFirstNum and tbSecondNum textBoxes and
        ///     finds their general common divisors.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        1 reference
        private void BtnFind_Click(object sender, EventArgs e)
        {
            try
            {
                int firstNum = Int32.Parse(tbFirstNum.Text);
                int secondNum = Int32.Parse(tbSecondNum.Text);

                lblGcd.Text = "Greatest Common Divisor : " + Gcd(firstNum, secondNum).ToString();
            }
            catch (Exception exception)
            {
                lblGcd.Text = exception.Message.ToString();
            }

        }

        /// <summary>
        ///     This method takes two integer parameters and finds their general common divisors recursively.
        /// </summary>
        /// <param name="x">First integer number</param>
        /// <param name="y">Second integer number</param>
        /// <returns>General common divisor</returns>
        2 references
        private int Gcd(int x, int y)
        {
            if (y == 0)
                return x;
            else
                return Gcd(y, x % y);
        }
    }
}
```

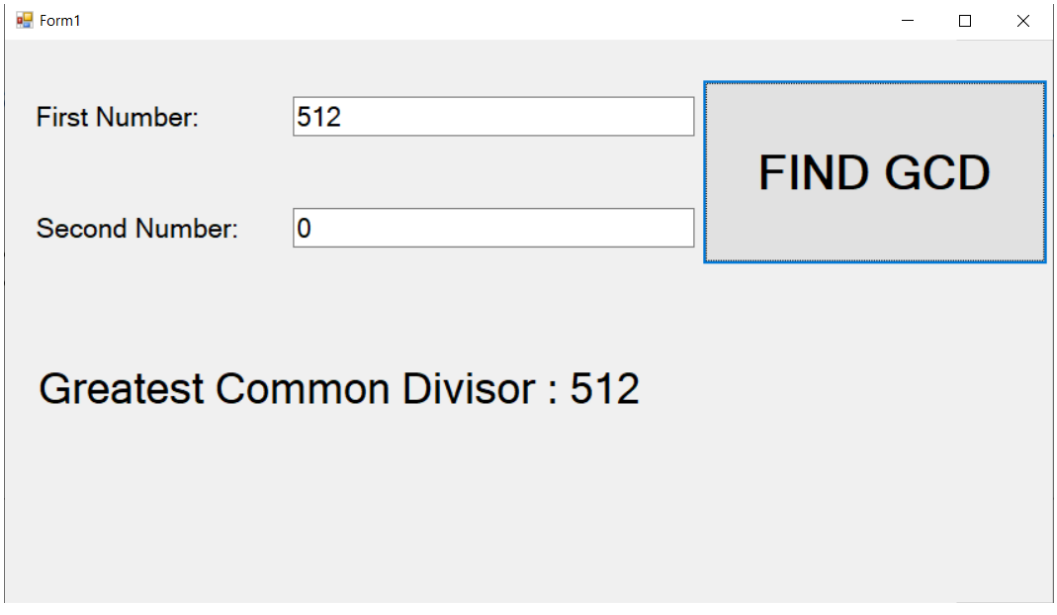
Outputs of program:

- Starting the application:

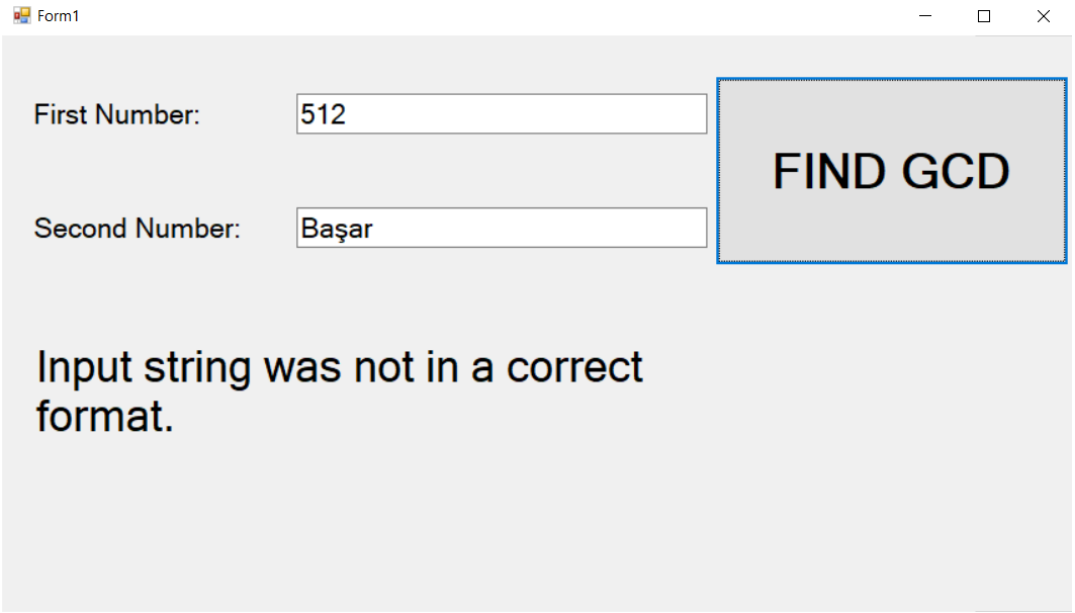


- Greatest Common Divisor finding examples:

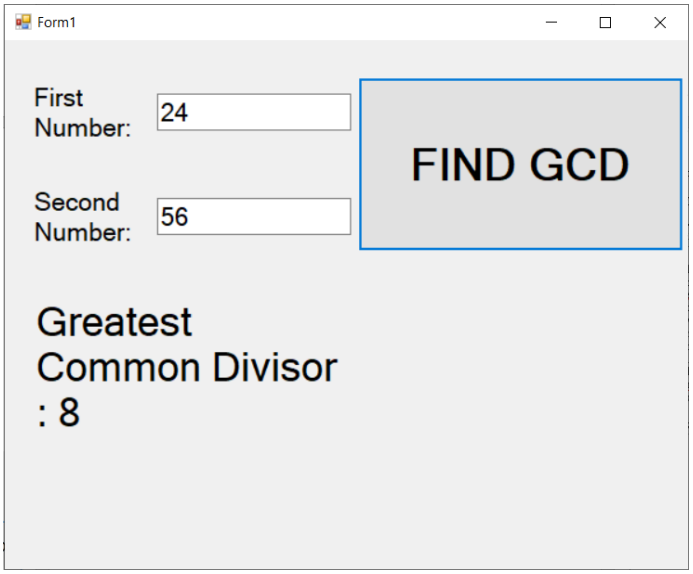


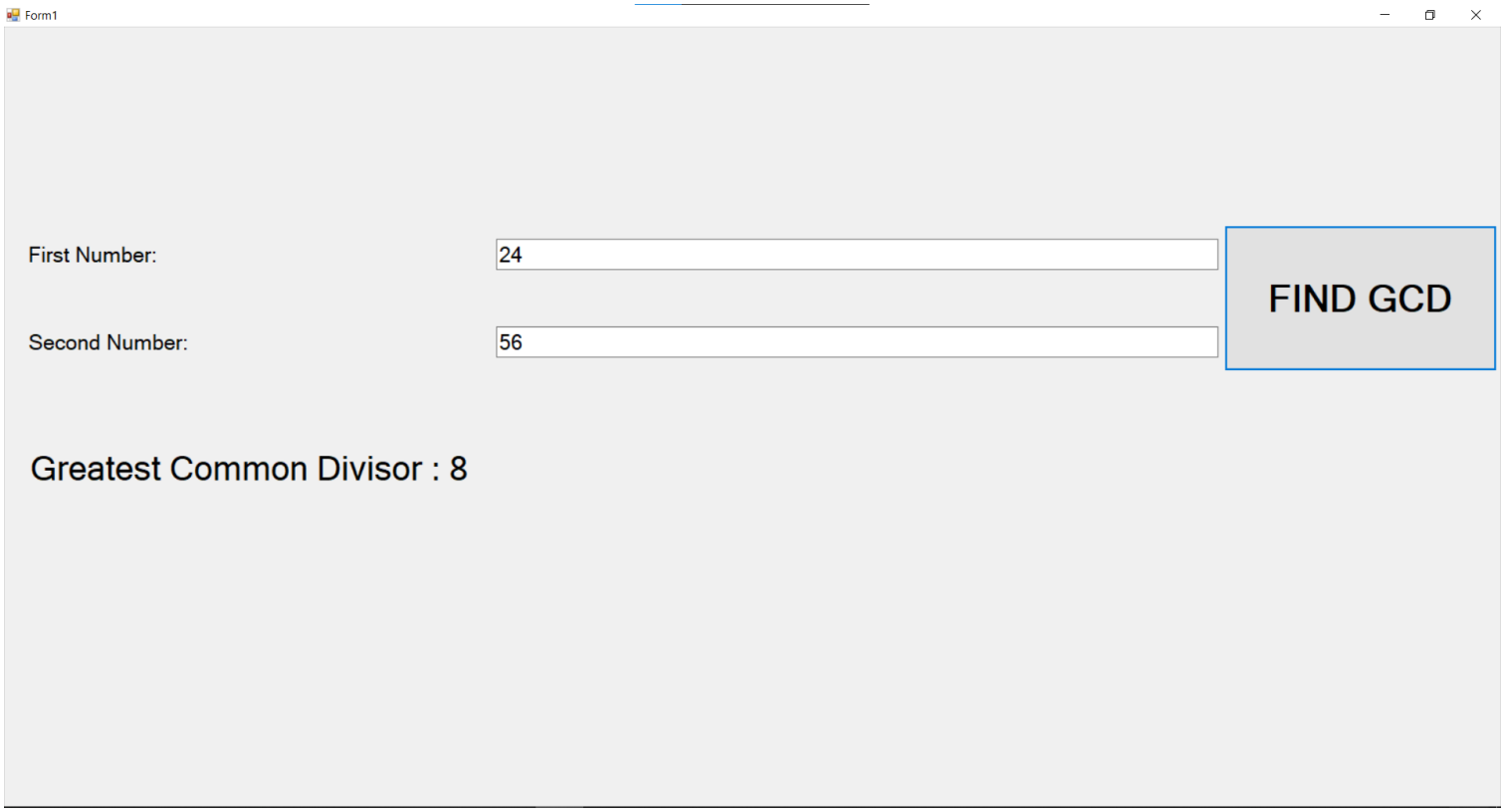


- Inputs must be integer that means they should be validated:



- Responsive design:




Form1
First Number:
24
Second Number:
56
FIND GCD
Greatest Common Divisor : 8

Question 7.9:

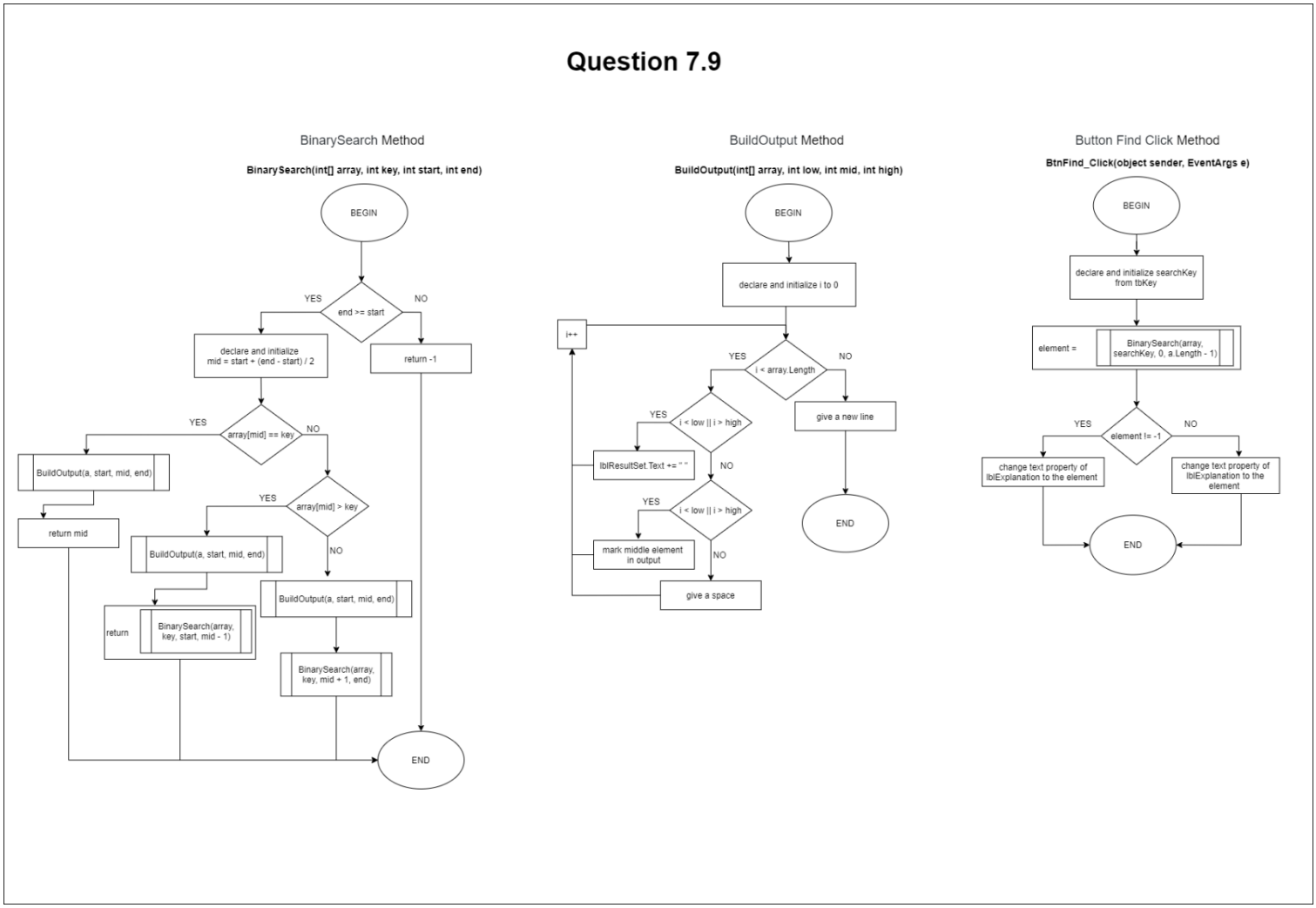
(Binary Search) Modify the program in Fig. 7.12 to use a recursive method BinarySearch to perform the binary search of the array. The method should receive an integer array and the starting and ending subscript as arguments. If the search key is found, return the array subscript; otherwise, return -1.

Solution 7.9:

PSEUDOCODE:

- User should enter a key to textBox to search in array
- If user clicks to the button, binary search algorithm should be called to find key on given array.
- In BinarySearch(int[] array, int key, int start, int end) method:
-- first we should check that end index is bigger than start index. If bigger, return -1 which means search key not found, otherwise continue.
--- If the element is present at the middle itself we should return the middle number
--- If element is smaller than mid, then it can only be present in left subarray (which means end <- mid -1), we should return BinarySearch(array, key, start, mid - 1)
--- Else the element can only be present in right subarray (which means start <- mid +1), we should return BinarySearch(array, key, mid + 1, end)
- If key is found, it should be shown in the screen, otherwise print that search key not found

FLOWCHART:



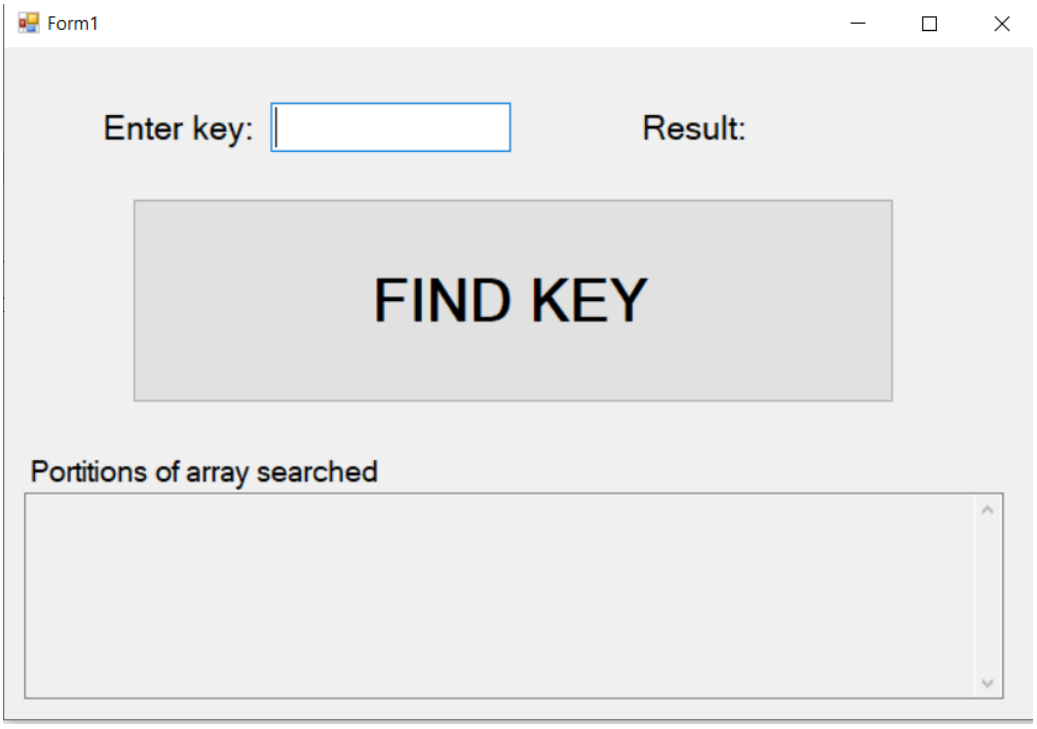
CODE:

```csharp
using System;

/**
 * 7.9 (Binary Search) Modify the program in Fig. 7.12 to use a recursive method BinarySearch to perform the binary search of the array.
 * The method should receive an integer array and the starting and ending subscript as arguments. If the search key is found,
 * return the array subscript; otherwise, return -1.
 */
namespace HW_7._9
{
    public partial class Form : System.Windows.Forms.Form
    {
        int[] a = { 0, 2, 4, 6, 8, 10, 12, 14, 16, 26, 18, 20, 22, 24, 26, 28 }; // array which will be searched to find key

        /// <summary>
        ///     Form initializer
        /// </summary>
        public Form()
        {
            InitializeComponent();
            lblSearched.Text = "Portitions of array searched";
        }

        /// <summary>
        ///     This method is listener of btnFind button. Calls Flip() method and sets other tools properties.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>

        private void BtnFind_Click(object sender, EventArgs e)
        {
            int searchKey;
            try
            {
                searchKey = Int32.Parse(tbKey.Text);
                lblResultSet.Text = "";


                // perform the binary search
                int element = BinarySearch(a, searchKey, 0, a.Length - 1);

                if (element != -1)
                    lblExplanation.Text = "Found value at element " + element;
                else
                    lblExplanation.Text = "Value not found";
            }
            catch (Exception ex)
            {
                lblResultSet.Text = ex.Message.ToUpperInvariant();
            }

        }


        /// <summary>
        ///     Binary Search method to find given key in given array that offered by book.
        /// </summary>
        /// <param name="array">array which will be searched to find key</param>
        /// <param name="key">element which will be searched in array</param>
        /// <returns></returns>
        public int BinarySearch(int[] array, int key)
        {
            int low = 0; // low subscript
            int high = array.Length - 1; // high subscript
            int middle; // middle subscript

            while (low <= high)
            {
                middle = (low + high) / 2;

                // the following line displays the portion
                // of the array currently being manipulated during
                // each iteration of the binary search loop
                BuildOutput(a, low, middle, high);

                if (key == array[middle]) // match
                    return middle;
                else if (key < array[middle])
                    high = middle - 1; // search low end of array
                else
                    low = middle + 1;

            } // end BinarySearch

            return -1; // search key not found

        }

        /// <summary>
        ///     Binary Search method to find given key in given array recursively.
        /// </summary>
        /// <param name="array">array which will be searched to find key</param>
        /// <param name="key">element which will be searched in array</param>
        /// <param name="start">first index to start searching in array</param>
        /// <param name="end">last index to end searching in array</param>
        /// <returns></returns>
        public int BinarySearch(int[] array, int key, int start, int end)
        {
```

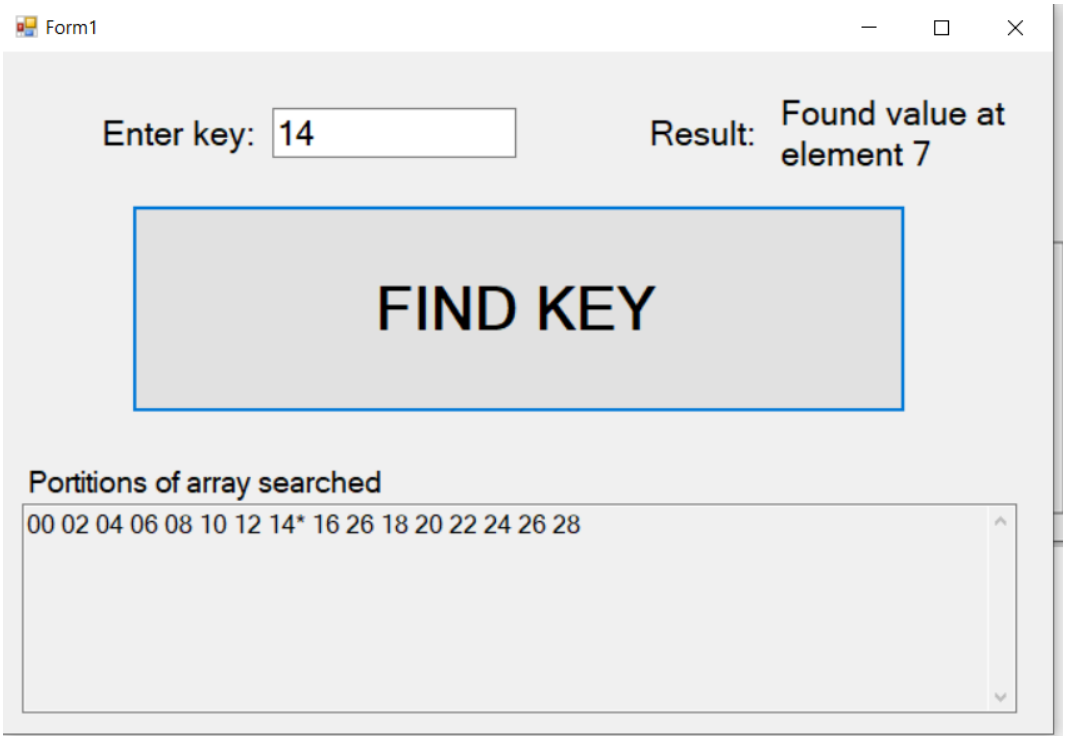
```
            if (end >= start)
            {
                int mid = start + (end - start) / 2;

                // If the element is present at the
                // middle itself
                if (array[mid] == key)
                {
                    BuildOutput(a, start, mid, end);
                    return mid;
                }


                // If element is smaller than mid, then
                // it can only be present in left subarray
                if (array[mid] > key)
                {
                    BuildOutput(a, start, mid, end);
                    return BinarySearch(array, key, start, mid - 1);

                }

                // Else the element can only be present
                // in right subarray
                BuildOutput(a, start, mid, end);
                return BinarySearch(array, key, mid + 1, end);
            }

            return -1; // search key not found

        }

        /// <summary>
        ///     This method prints the search steps in lblResultSet tool.
        /// </summary>
        /// <param name="array">array which is searched to find key</param>
        /// <param name="low">low subscript</param>
        /// <param name="mid">middle subscript</param>
        /// <param name="high">high subscript</param>
        4 references
        public void BuildOutput(int[] array, int low, int mid, int high)
        {
            for (int i = 0; i < array.Length; i++)
            {
                if (i < low || i > high)
                    lblResultSet.Text += "  ";

                // mark middle element in output
                else if (i == mid)
                    lblResultSet.Text += array[i].ToString("00") + "* ";
                else
                    lblResultSet.Text += array[i].ToString("00") + " ";
            }

            lblResultSet.Text += "\r\n";

        } // end BuildOutput


    }
}
```

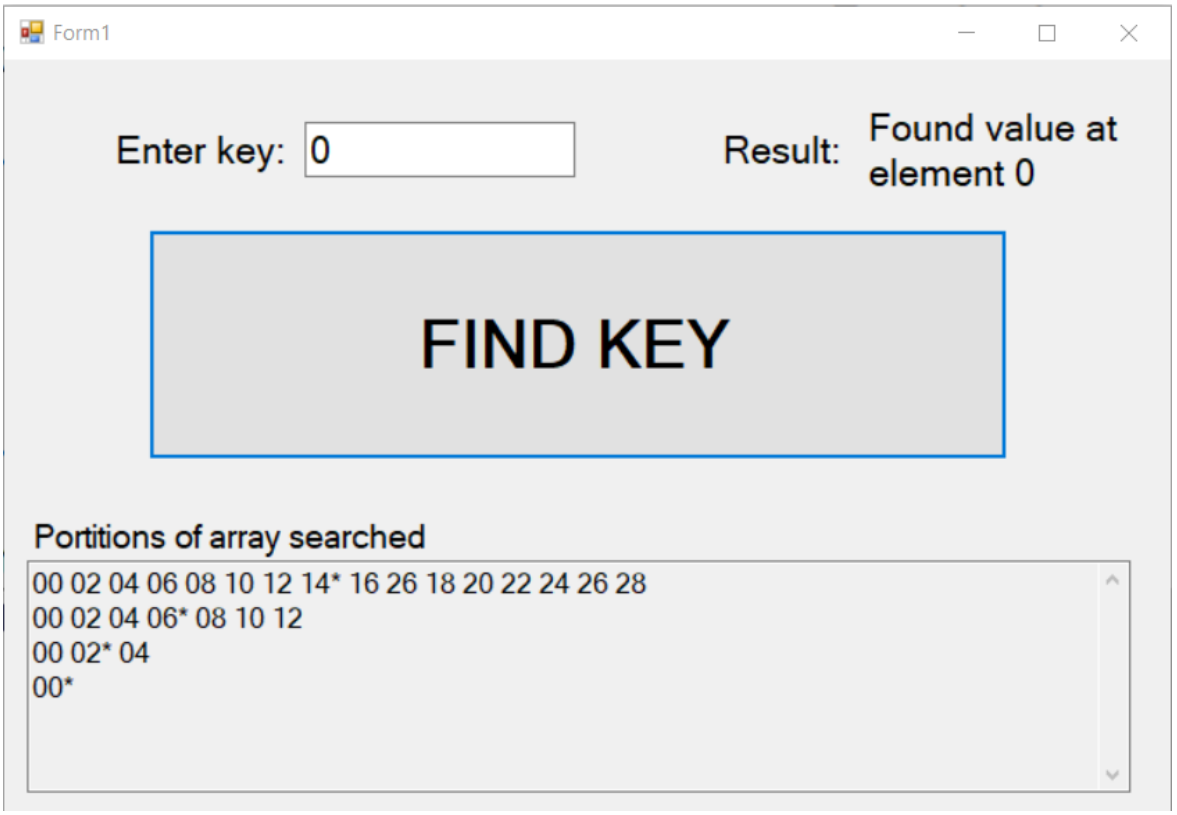
Outputs of program:

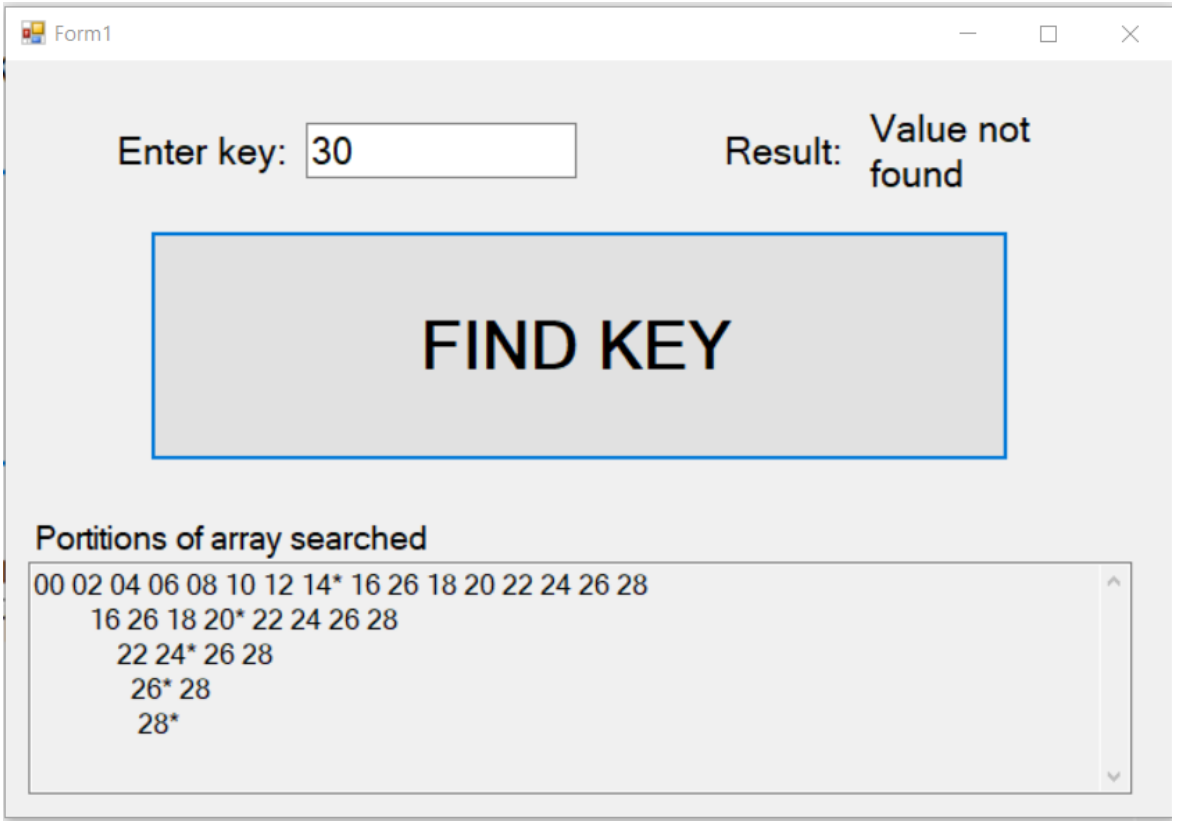
- Starting the application:



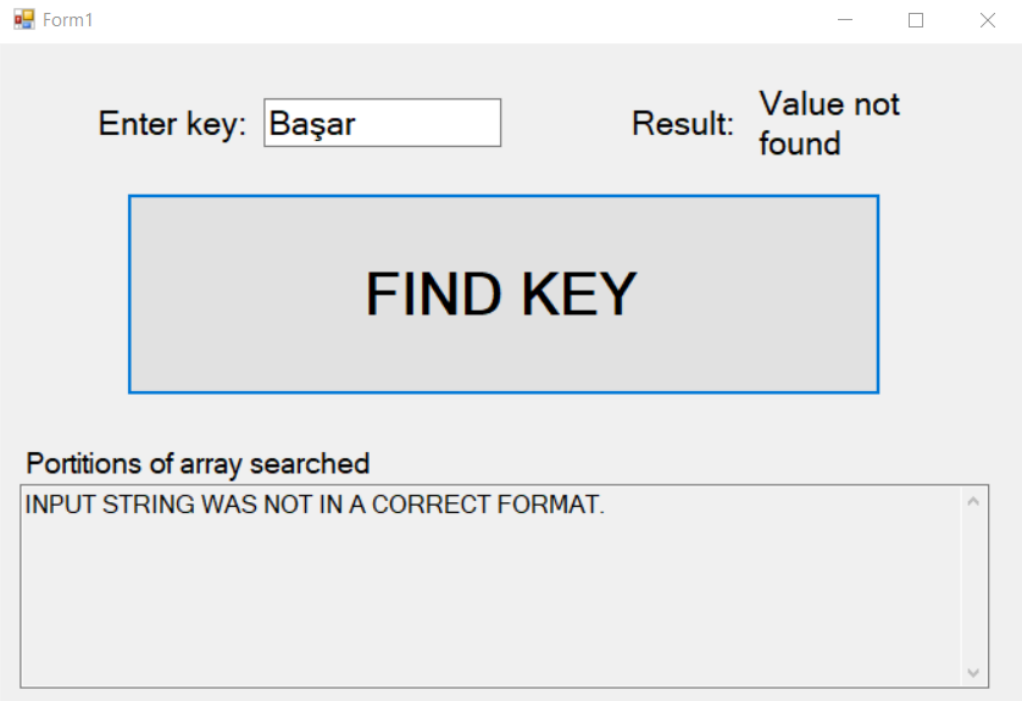
- Binary Search examples:



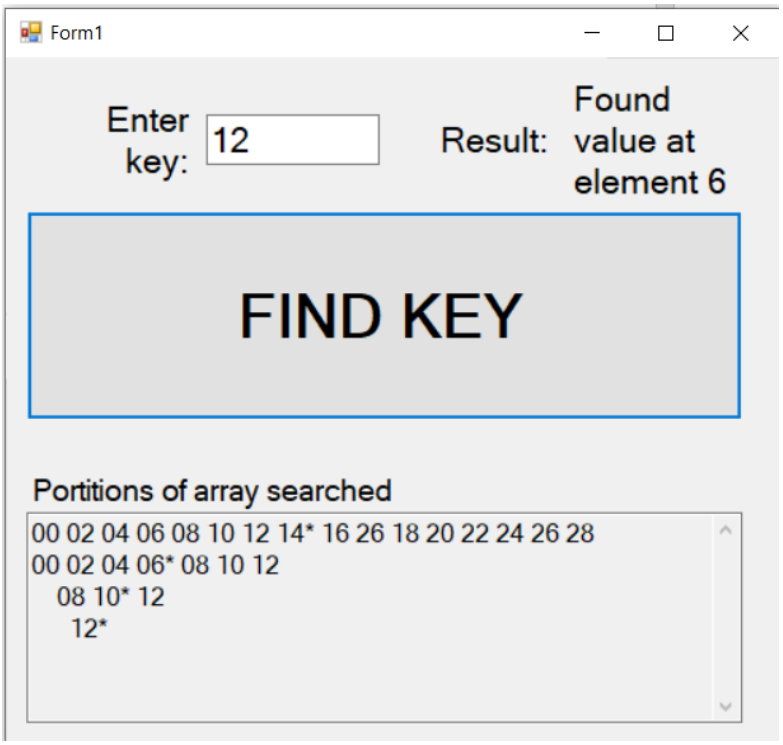

Form1
Enter key: 0
Result: Found value at element 0
FIND KEY
Portitions of array searched
00 02 04 06 08 10 12 14* 16 26 18 20 22 24 26 28
00 02 04 06* 08 10 12
00 02* 04
00*

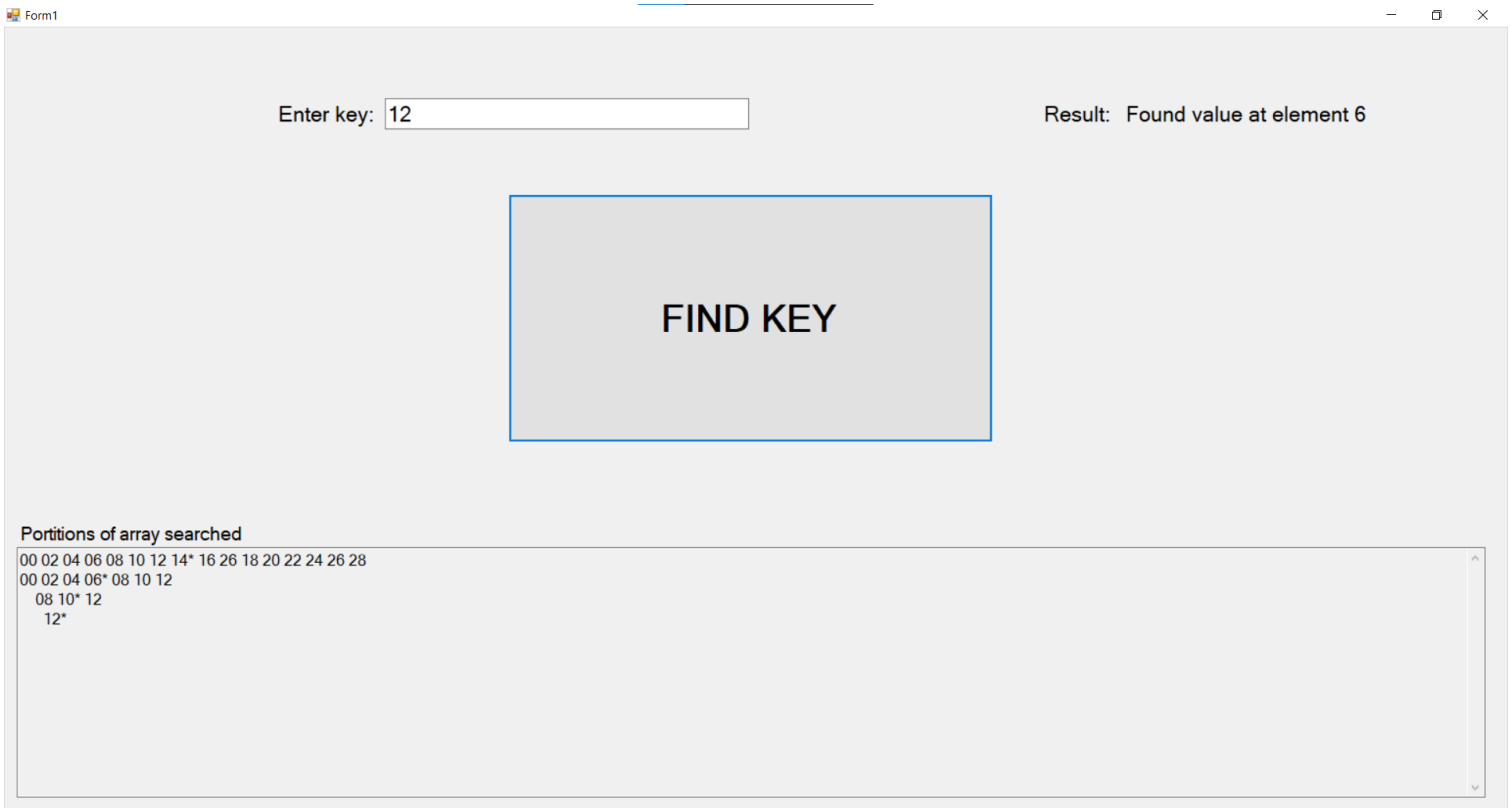

Form1
Enter key: 30
Result: Value not found
FIND KEY
Portitions of array searched
00 02 04 06 08 10 12 14* 16 26 18 20 22 24 26 28
16 26 18 20* 22 24 26 28
22 24* 26 28
26* 28
28*

- Input must be integer that means it should be validated:



- Responsive design:




Form1
Enter key: 12
Result: Found value at element 6
FIND KEY
Portitions of array searched
00 02 04 06 08 10 12 14* 16 26 18 20 22 24 26 28
00 02 04 06* 08 10 12
08 10* 12
12*