

### Question 3.8:

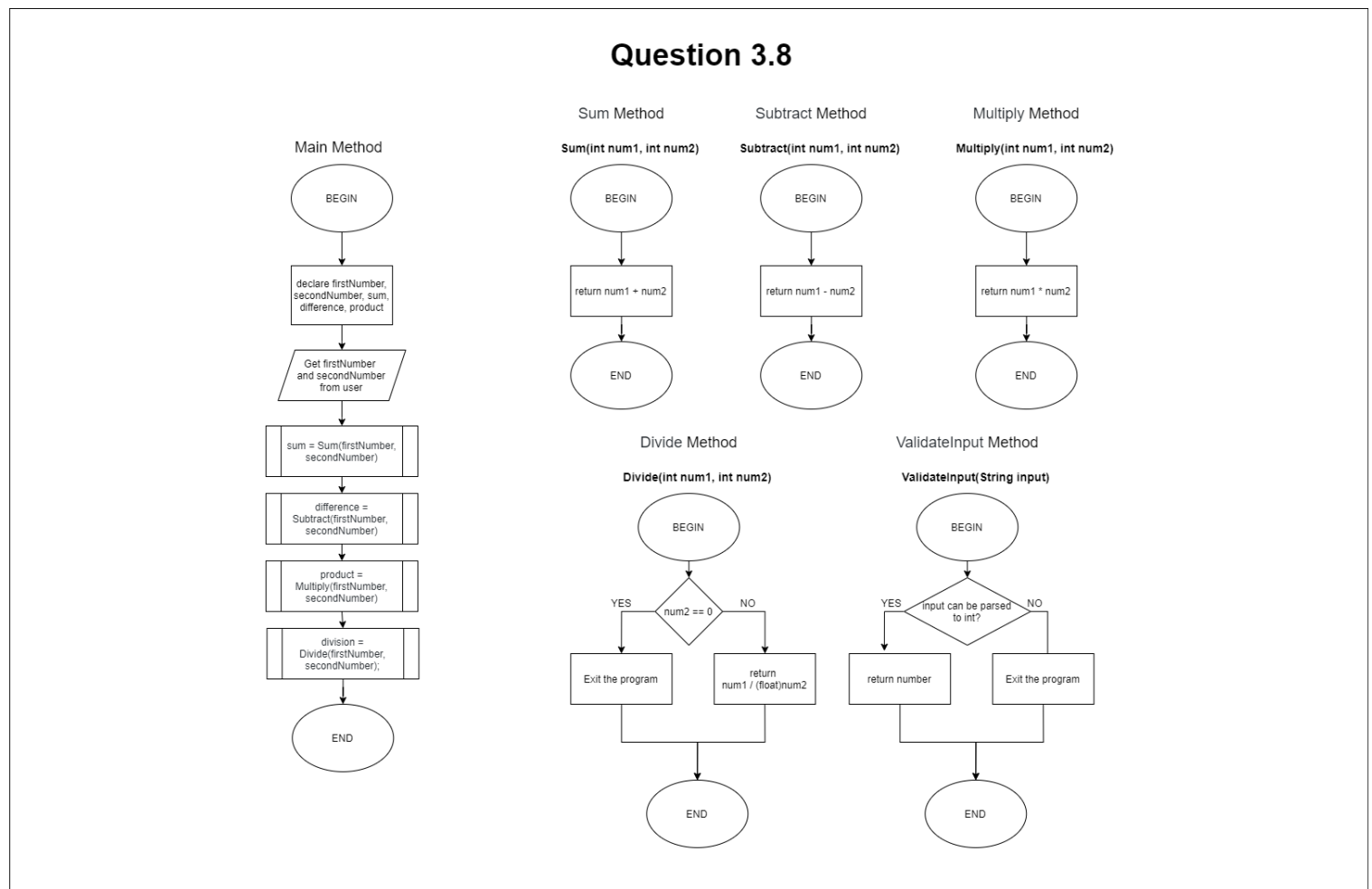
Write an application that asks the user to enter two numbers, obtains the two numbers from the user and prints the sum, product, difference and quotient of the two numbers.

### Solution 3.8:

#### PSEUDOCODE:

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

#### FLOWCHART:



## CODE:

```
HW_3.8 HW_3.8.Program Main(string[] args)
1 using System;
2
3
4 /*
5  * 3.8 Write an application that asks the user to enter two numbers, obtains the two numbers from
6  * the user and prints the sum, product, difference and quotient of the two numbers.
7  */
8 namespace HW_3_8
9 {
10     class Program
11     {
12         // main entry point for the application
13         static void Main(string[] args)
14         {
15             int firstNumber, secondNumber, sum, difference, product;
16             float division;
17
18             String description = "Application that obtains the two numbers from the user and prints the sum, product, difference and quotient of the two numbers: \n";
19             Console.WriteLine(description.ToUpperInvariant());
20
21             // requesting two numbers from user to calculate summation, subtraction, difference and division
22             // also inputs will be checked with ValidateInput() method to ensure that inputs are numbers
23             Console.WriteLine("Please enter the first number:");
24             firstNumber = ValidateInput(Console.ReadLine());
25
26             Console.WriteLine("Please enter the second number:");
27             secondNumber = ValidateInput(Console.ReadLine());
28
29             sum = Sum(firstNumber, secondNumber);
30             difference = Subtract(firstNumber, secondNumber);
31             product = Multiply(firstNumber, secondNumber);
32             division = Divide(firstNumber, secondNumber);
33
34             Console.WriteLine("Sum \t\t: " + sum + "\nDifference \t: " + difference + "\nProduct \t: " + product +
35                             "\nDivision \t: " + division);
36
37             Console.WriteLine("Press any key to close application");
38             Console.ReadKey();
39         }
40
41         /// <summary>
42         /// This method takes two integer parameters and returns the sum of them.
43         /// </summary>
44         /// <param name="num1">First integer number to sum</param>
45         /// <param name="num2">Second integer number to sum</param>
46         /// <returns>The summation of two doubles</returns>
47         static int Sum(int num1, int num2)
48         {
49             return num1 + num2;
50         }
51
52         /// <summary>
53         /// This method takes two integer parameters and returns the subtraction of them.
54         /// </summary>
55         /// <param name="num1">First integer number to subtract</param>
56         /// <param name="num2">Second integer number to subtract</param>
57         /// <returns>The subtraction of two doubles</returns>
58         static int Subtract(int num1, int num2)
59         {
60             return num1 - num2;
61         }
62
63         /// <summary>
64         /// This method takes two integer parameters and returns the multiplication of them.
65         /// </summary>
66         /// <param name="num1">First integer number to multiply</param>
67         /// <param name="num2">Second integer number to multiply</param>
68         /// <returns>The multiplication of two doubles</returns>
69         static int Multiply(int num1, int num2)
70         {
71             return num1 * num2;
72         }
73
74         /// <summary>
75         /// This method takes two integer parameters and returns the division of them.
76         /// </summary>
77         /// <param name="num1">First integer number which is dividend</param>
78         /// <param name="num2">Second integer number which is divisor</param>
79         /// <returns>The division of two integer number in float type</returns>
80         static float Divide(int num1, int num2)
81         {
82             // divisor can not be zero!
83             if (num2 == 0)
84             {
85                 Console.Write("Divisor is 0 and any number can not be divided by 0!! \nPress any key to Exit the program!");
86                 Console.ReadKey();
87                 Environment.Exit(0);
88                 return 0;
89             }
90             else
91             {
92                 return (num1 / (float)num2);
93             }
94         }
95
96         /// <summary>
97         /// This method takes an string parameter and check if the content is integer or not
98         /// </summary>
99         /// <param name="input">User input</param>
100
101
```

```

102     /// <returns></returns>
103     static int ValidateInput(String input)
104     {
105         int number;
106         // if input is parseable, return as integer
107         if (Int32.TryParse(input, out number))
108         {
109             return number;
110         }
111         else
112         {
113             Console.WriteLine("Input is not integer!! \nPress any key to Exit the program!");
114             Console.ReadKey();
115             Environment.Exit(0);
116             return 0;
117         }
118     }
119 }
120
121

```

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:

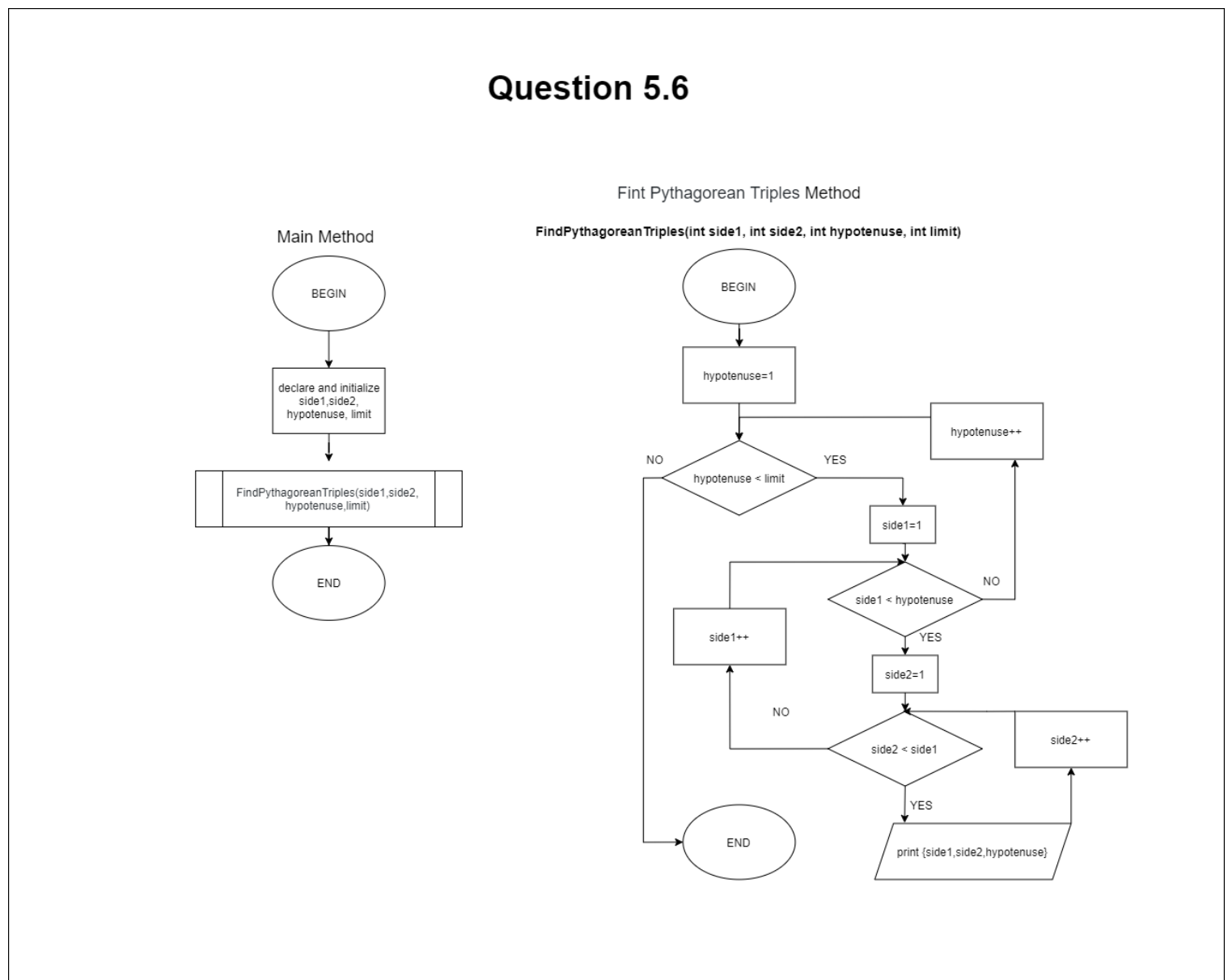
-Pythagorean triples rules:

- side1 or side2 or hypotenuse can not be bigger than upper limit
- $\text{side1}^2 + \text{side2}^2 = \text{hypotenuse}^2$  (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)
1 using System;
2
3
4 /*
5  * 5.6 (Pythagorean Triples) A right triangle can have sides that are all integers.
6  * A set of three integer values for the sides of a right triangle is called a Pythagorean triple.
7  * These three sides must satisfy the relationship that the sum of the squares of the two sides is equal to
8  * the square of the hypotenuse. Write a program to find all Pythagorean triples for side1, side2 and hypotenuse,
9  * none larger than 30. Use a triple-nested for loop that tries all possibilities. This is an example of
10  * "brute force" computing. You will learn in more advanced computer science courses that there are
11  * several problems for which there is no other known algorithmic approach.
12  */
13 namespace HW_5_6
14 {
15     0 references
16     class Program
17     {
18         // main entry point for the application
19         0 references
20         static void Main(string[] args)
21         {
22             int side1 = 1, side2 = 1, hypotenuse = 1;
23             int limit = 30;
24
25             String description = "Application that find all Pythagorean triples for side1, side2 and hypotenuse, none larger than " + limit + " \n";
26             Console.WriteLine(description.ToUpperInvariant());
27
28             FindPythagoreanTriples(side1, side2, hypotenuse, limit);
29
30             Console.WriteLine("Press any key to close application");
31             Console.ReadKey();
32         }
33
34         1 reference
35         private static void FindPythagoreanTriples(int side1, int side2, int hypotenuse, int limit)
36         {
37             int counter = 1;
38             // we should try all possibilities to get all Pythagorean triples
39             // rules:
40             // - side1 or side2 or hypotenuse can not be bigger than 30
41             // - side1^2 + side2^2 = hypotenuse^2
42             for (hypotenuse = 1; hypotenuse < limit; hypotenuse++)
43             {
44                 for (side1 = 1; side1 < hypotenuse; side1++)
45                 {
46                     for (side2 = 1; side2 < side1; side2++)
47                     {
48                         if (Math.Abs(Math.Pow(side1, 2) + Math.Pow(side2, 2) - Math.Pow(hypotenuse, 2)) < 0.001)
49                         {
50                             Console.WriteLine(
51                                 counter + "\t{" + side2 + ", " + side1 + ", " + hypotenuse + "}");
52                             counter++;
53                         }
54                     }
55                 }
56             }
57         }
58     }
59 }
```

Outputs of program:

```
C:\Users\YARGIC\Desktop\Visual\Midterm\HW_5.6\bin\Debug\HW_5.6.exe
APPLICATION THAT FIND ALL PYTHAGOREAN TRIPLES FOR SIDE1, SIDE2 AND HYPOTENUSE, NONE LARGER THAN 30
1      {3,4,5}
2      {6,8,10}
3      {5,12,13}
4      {9,12,15}
5      {8,15,17}
6      {12,16,20}
7      {15,20,25}
8      {7,24,25}
9      {10,24,26}
10     {20,21,29}
Press any key to close application
```

```
C:\Users\YARGIC\Desktop\Visual\Midterm\HW_5.6\bin\Debug\HW_5.6.exe
APPLICATION THAT FIND ALL PYTHAGOREAN TRIPLES FOR SIDE1, SIDE2 AND HYPOTENUSE, NONE LARGER THAN 50
1      {3,4,5}
2      {6,8,10}
3      {5,12,13}
4      {9,12,15}
5      {8,15,17}
6      {12,16,20}
7      {15,20,25}
8      {7,24,25}
9      {10,24,26}
10     {20,21,29}
11     {18,24,30}
12     {16,30,34}
13     {21,28,35}
14     {12,35,37}
15     {15,36,39}
16     {24,32,40}
17     {9,40,41}
18     {27,36,45}
Press any key to close application
```

### Question 6.13:

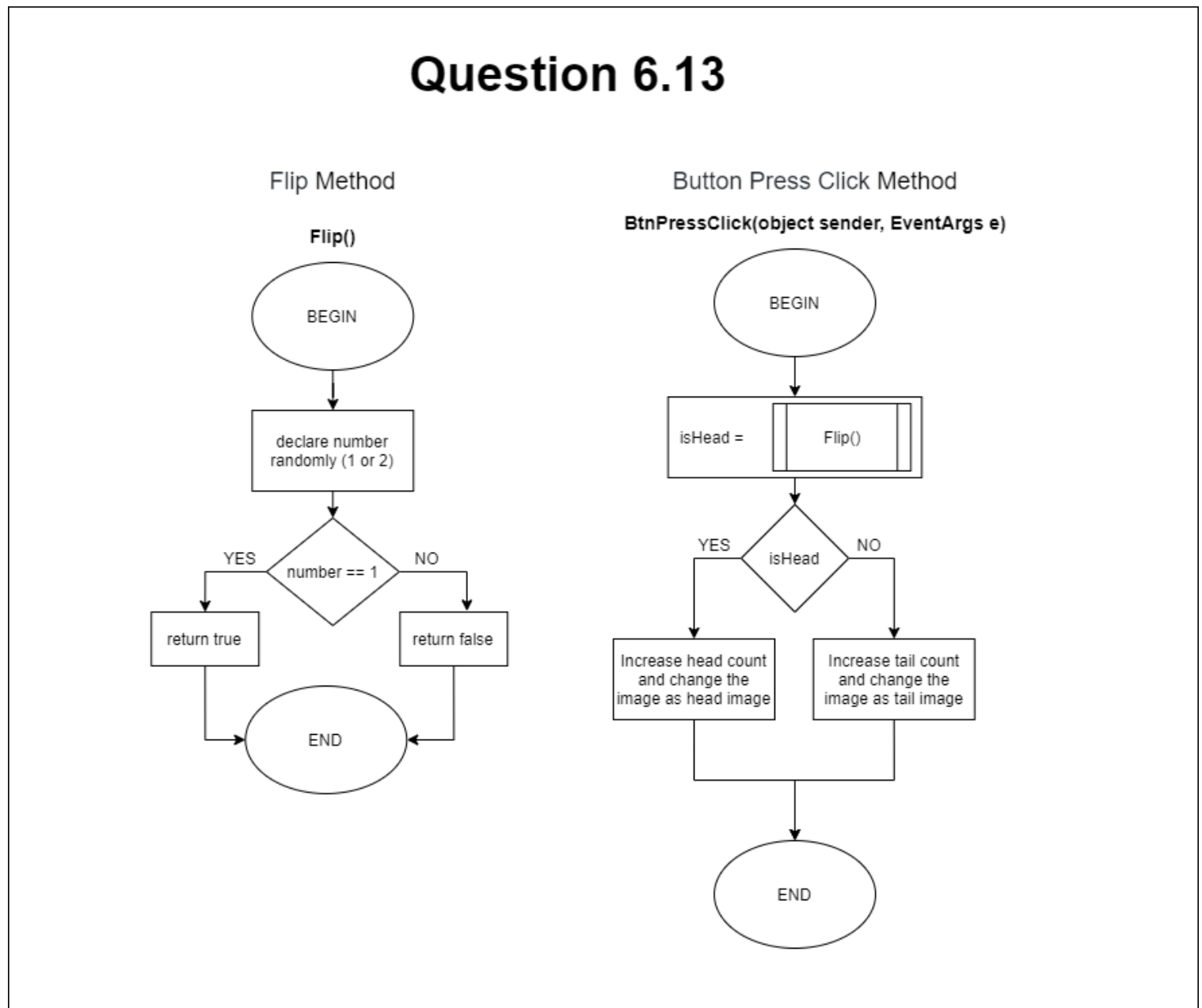
Write an application that simulates coin tossing. Let the program toss the coin each time the user presses the “Toss” button. Count the number of times each side of the coin appears. Display the results. The program should call a separate method Flip that takes no arguments and returns false for tails 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
1 using System;
2 using System.Drawing;
3 using System.Windows.Forms;
4
5 /**
6  *
7  * 6.13 Write an application that simulates coin tossing. Let the program toss the coin each time the
8  * user presses the "Toss" button. Count the number of times each side of the coin appears. Display the
9  * results. The program should call a separate method Flip that takes no arguments and returns false
10  * for tails and true for heads. [Note: If the program realistically simulates the coin tossing, each side
11  * of the coin should appear approximately half of the time.]
12  *
13  */
14
15 namespace HW_6._13
16 {
17     3 references
18     public partial class Form1 : Form
19     {
20         Random random = new Random(); // random instance to generate random number
21         string txt = "";
22
23         /// <summary>
24         ///     Form initializer
25         /// </summary>
26         1 reference
27         public Form1()
28         {
29             InitializeComponent();
30         }
31
32         /// <summary>
33         ///     This method generates random number 1 or 2.
34         ///     1 means head, 2 means tail.
35         /// </summary>
36         /// <returns>true if result is head</returns>
37         1 reference
38         private bool Flip()
39         {
40             int number = random.Next(1, 3); // returns either 1 or 2
41             //head
42             if (number == 1) return true;
43             //tail
44             return false;
45         }
46
47         /// <summary>
48         ///     This method is listener of btnPress button. Calls Flip() method and sets other tools properties.
49         /// </summary>
50         /// <param name="sender"></param>
51         /// <param name="e"></param>
52         1 reference
53         private void BtnPressClick(object sender, EventArgs e)
54         {
55             // Flip() returns true if side of coin is head
56             bool isHead = Flip();
57
58             if (isHead)
59             {
60                 lblHeadCount.Text = (int.Parse(lblHeadCount.Text) + 1).ToString(); // increasing count
61                 txt = "HEAD"; // will be used for changing the text of lblText
62                 pictureBox.Image = new Bitmap(Properties.Resources.head); // changing the image of pictureBox
63             }
64             else
65             {
66                 lblTailCount.Text = (int.Parse(lblTailCount.Text) + 1).ToString();
67                 txt = "TAIL";
68                 pictureBox.Image = new Bitmap(Properties.Resources.tail);
69             }
70             lblText.Text = txt;
71         }
72     }
73 }
```



Outputs of program:

- Starting the application:

Form1

Tail Count 0

Head Count 0

Toss

- After toss button click:

Form1

HEAD


Tail Count 4

Head Count 2

Toss

- Toss again:

Form1



TAIL


Tail Count 5

Head Count 2

Toss

- Responsive design:

Form1



TAIL

Tail Count 5

Head Count 2

Toss



TAIL

Tail Count

5

Toss

Head Count

2

## Question 6.16:

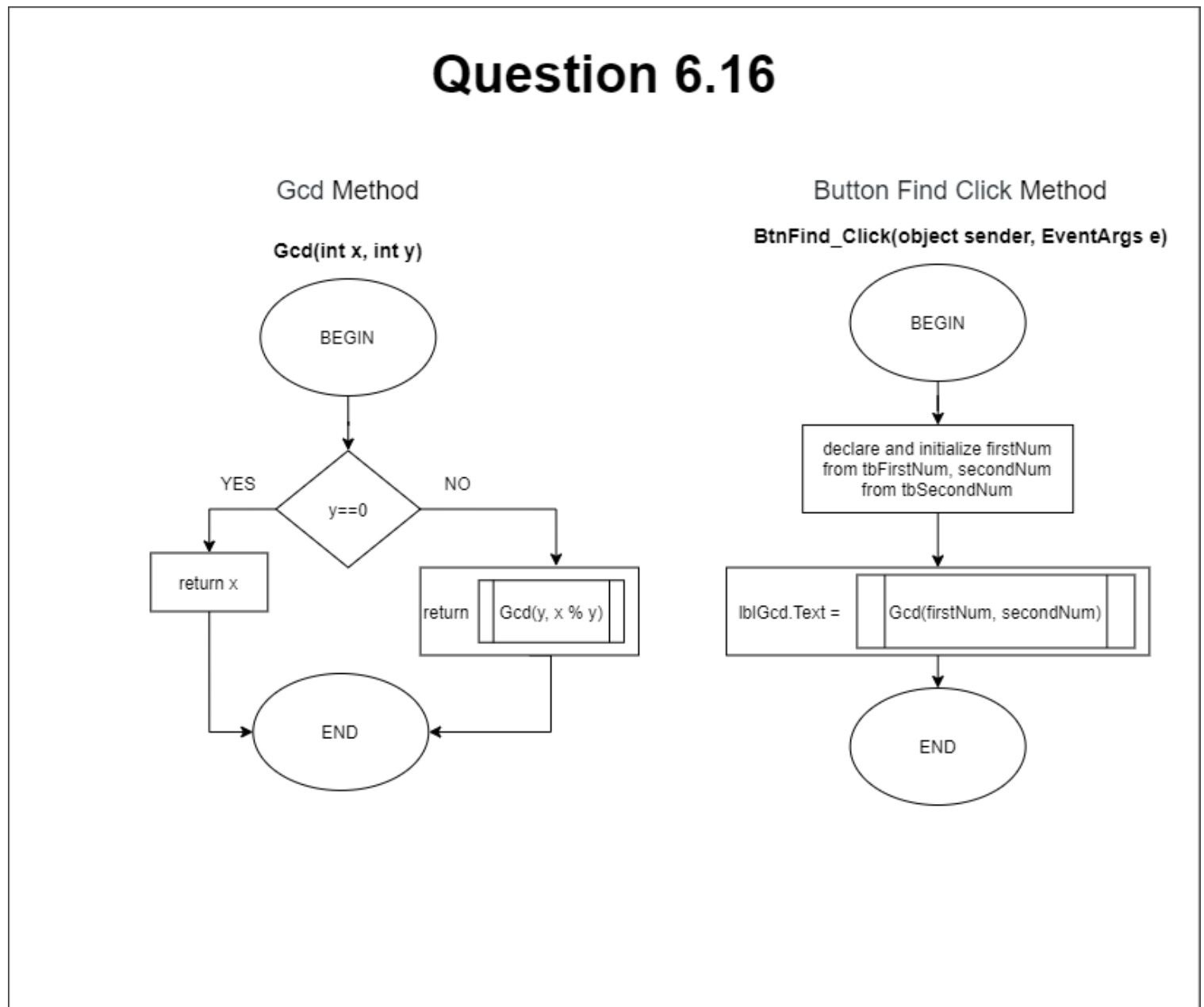
The greatest common divisor of integers  $x$  and  $y$  is the largest integer that evenly divides both  $x$  and  $y$ . Write a recursive method `Gcd` that returns the greatest common divisor of  $x$  and  $y$ . The Gcd of  $x$  and  $y$  is defined recursively as follows: If  $y$  is equal to 0, then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

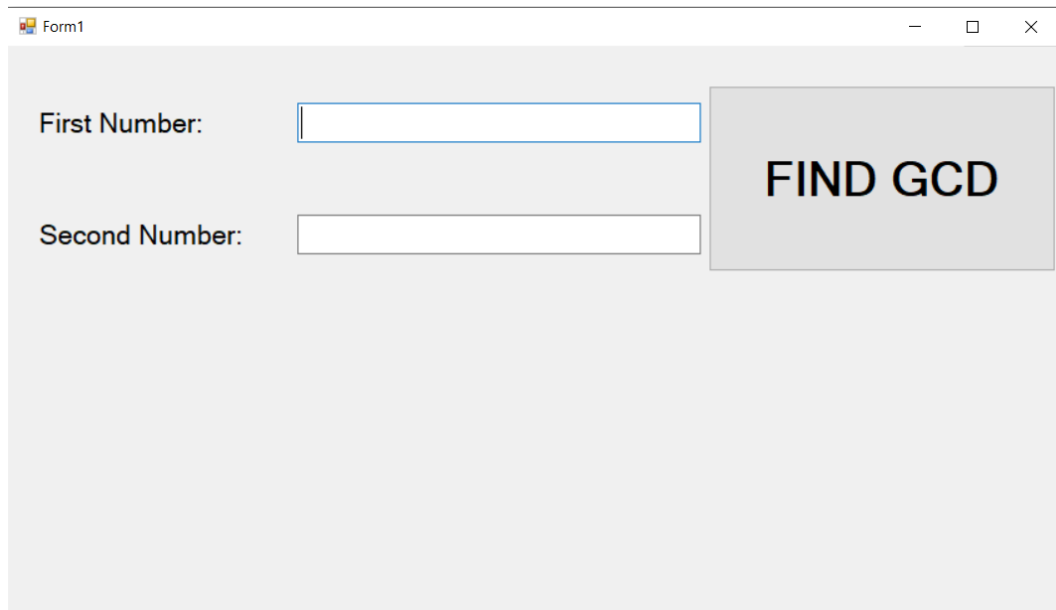


CODE:

```
HW_6.16 HW_6.16.Form1 tipUpper
1 using System;
2 using System.Windows.Forms;
3
4 /**
5  *
6  * 6.16 The greatest common divisor of integers x and y is the largest integer that evenly divides
7  * both x and y. Write a recursive method Gcd that returns the greatest common divisor of x and y. The
8  * Gcd of x and y is defined recursively as follows: If y is equal to 0, then Gcd( x, y ) is x; otherwise,
9  * Gcd( x, y ) is Gcd( y, x % y ), where % is the modulus operator.
10  *
11  */
12 namespace HW_6_16
13 {
14     3 references
15     public partial class Form1 : Form
16     {
17         /// <summary>
18         ///     Form initializer
19         /// </summary>
20         1 reference
21         public Form1()
22         {
23             InitializeComponent();
24         }
25
26         /// <summary>
27         ///     This method is listener of btnFind button. Gets two input from tbFirstNum and tbSecondNum textBoxes and
28         ///     finds their general common divisors.
29         /// </summary>
30         /// <param name="sender"></param>
31         /// <param name="e"></param>
32         1 reference
33         private void BtnFind_Click(object sender, EventArgs e)
34         {
35             try
36             {
37                 int firstNum = Int32.Parse(tbFirstNum.Text);
38                 int secondNum = Int32.Parse(tbSecondNum.Text);
39
40                 lblGcd.Text = "Greatest Common Divisor : " + Gcd(firstNum, secondNum).ToString();
41             }
42             catch (Exception exception)
43             {
44                 lblGcd.Text = exception.Message.ToString();
45             }
46         }
47
48         /// <summary>
49         ///     This method takes two integer parameters and finds their general common divisors recursively.
50         /// </summary>
51         /// <param name="x">First integer number</param>
52         /// <param name="y">Second integer number</param>
53         /// <returns>General common divisor</returns>
54         2 references
55         private int Gcd(int x, int y)
56         {
57             if (y == 0)
58                 return x;
59             else
60                 return Gcd(y, x % y);
61         }
62     }
63 }
```

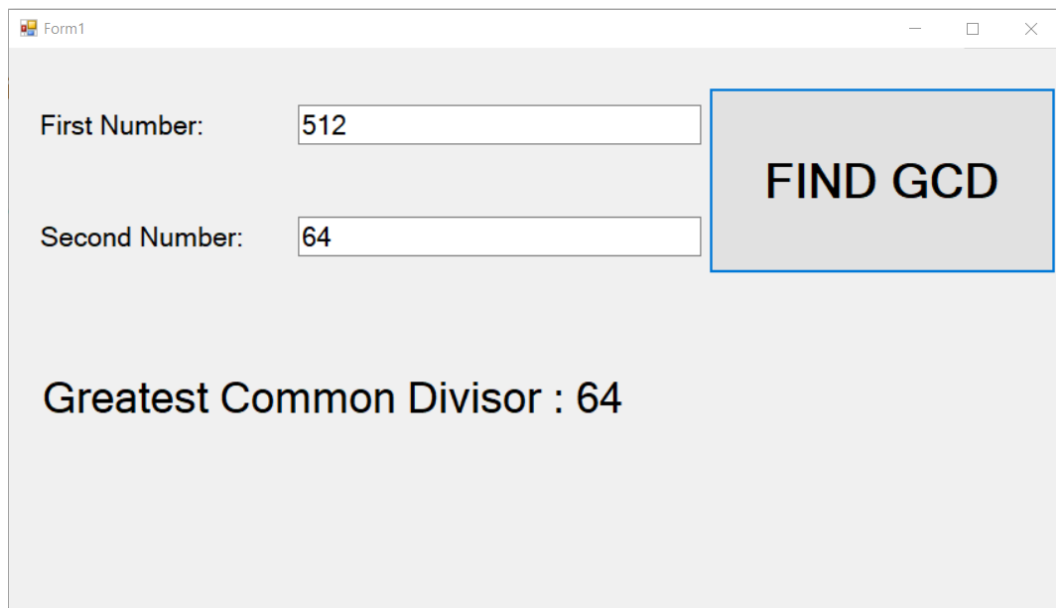
Outputs of program:

- Starting the application:

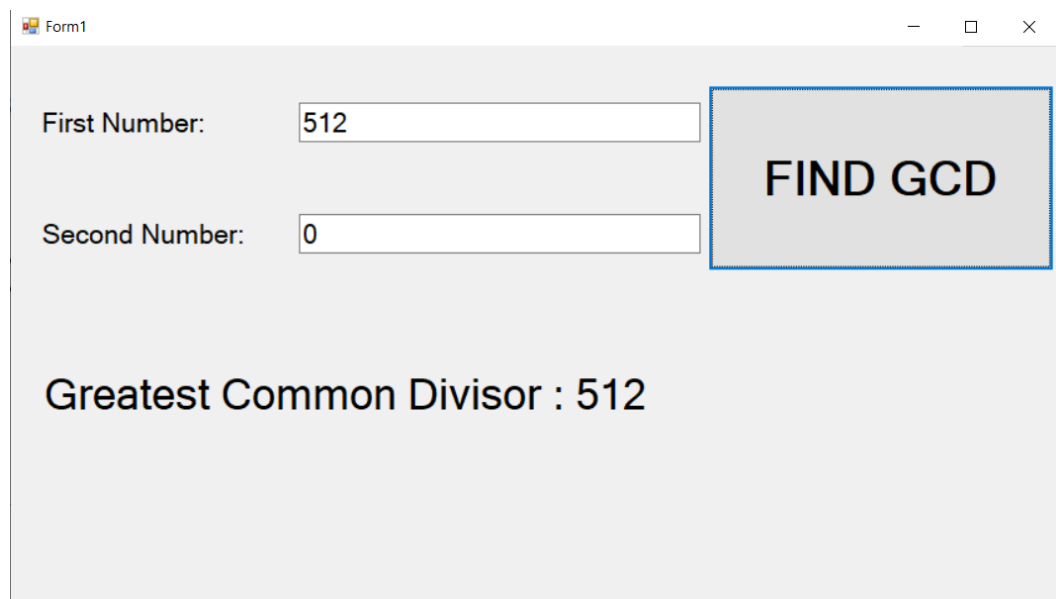


A screenshot of a Windows application window titled "Form1". The window has a light gray background. On the left, there are two labels: "First Number:" and "Second Number:". Next to each label is a white text input field. To the right of these fields is a gray rectangular button with the text "FIND GCD" in bold black letters. The input fields are currently empty.

- Greatest Common Divisor finding examples:

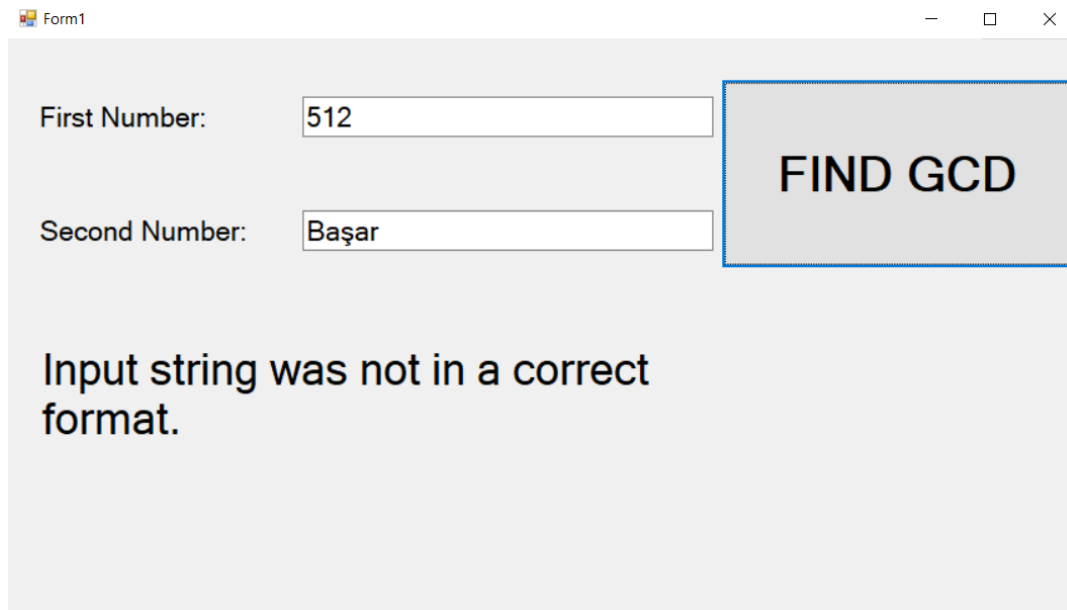


A screenshot of the "Form1" application window. The "First Number" input field contains the value "512" and the "Second Number" input field contains the value "64". The "FIND GCD" button is highlighted with a blue border. Below the input fields, the text "Greatest Common Divisor : 64" is displayed in a large black font.



A screenshot of the "Form1" application window. The "First Number" input field contains the value "512" and the "Second Number" input field contains the value "0". The "FIND GCD" button is highlighted with a blue border. Below the input fields, the text "Greatest Common Divisor : 512" is displayed in a large black font.

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



Form1

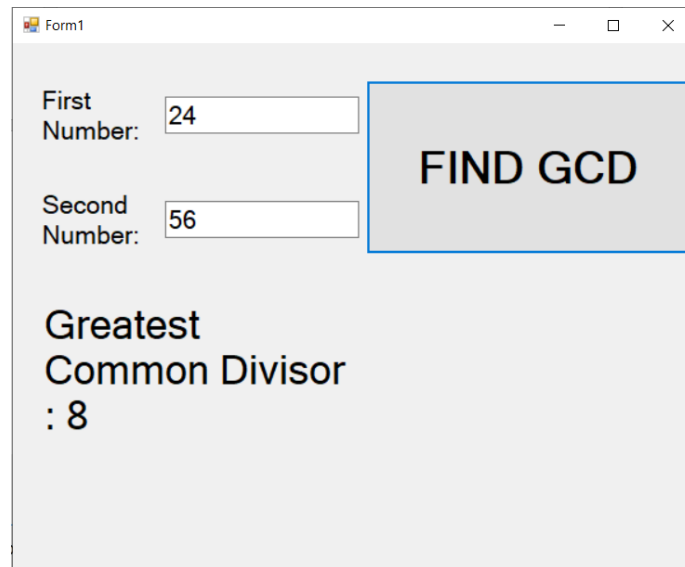
First Number: 512

Second Number: Başar

**FIND GCD**

Input string was not in a correct format.

- Responsive design:



Form1

First Number: 24

Second Number: 56

**FIND GCD**

Greatest  
Common Divisor  
: 8

First Number:

Second Number:

**FIND GCD**

Greatest Common Divisor : 8



## Question 7.9:

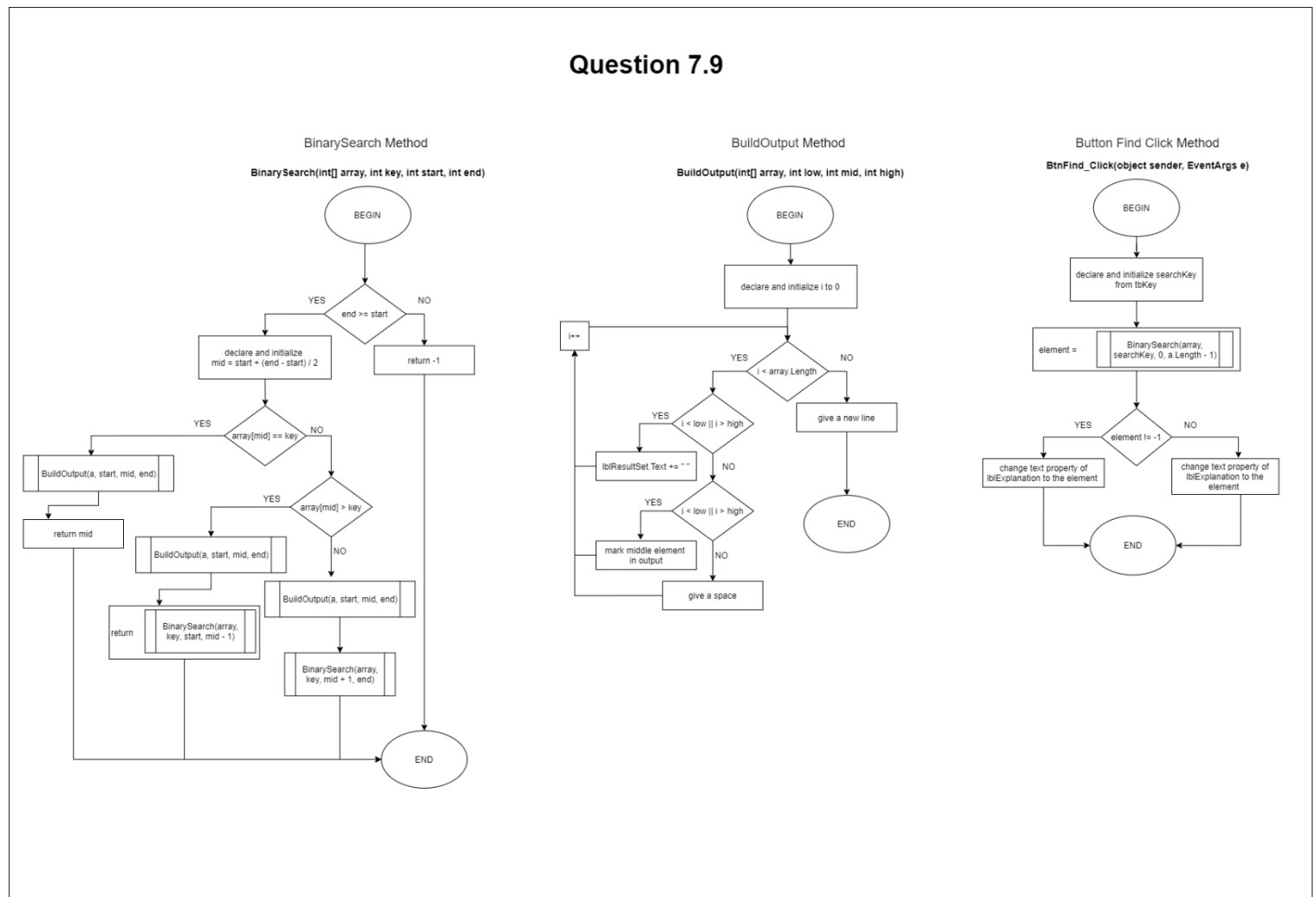
(Binary Search) Modify the program in Fig. 7.12 to use a recursive method BinarySearch to perform the binary search of the array. The method should receive an integer array and the starting and ending subscript as arguments. If the search key is found, return the array subscript; otherwise, return -1.

## 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  $\text{end} \leftarrow \text{mid} - 1$ ), we should return BinarySearch(array, key, start, mid - 1)
  - Else the element can only be present in right subarray (which means  $\text{start} \leftarrow \text{mid} + 1$ ), 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:

```

1  using System;
2
3  /**
4   * 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.
5   * The method should receive an integer array and the starting and ending subscript as arguments. If the search key is found,
6   * return the array subscript; otherwise, return -1.
7   */
8  namespace HW_7_9
9  {
10     3 references
11     public partial class Form : System.Windows.Forms.Form
12     {
13         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
14
15         /// <summary>
16         ///     Form initializer
17         /// </summary>
18         1 reference
19         public Form()
20         {
21             InitializeComponent();
22             lblSearched.Text = "Portitions of array searched";
23         }
24
25         /// <summary>
26         ///     This method is listener of btnFind button. Calls Flip() method and sets other tools properties.
27         /// </summary>
28         /// <param name="sender"></param>
29         /// <param name="e"></param>
30
31         1 reference
32         private void BtnFind_Click(object sender, EventArgs e)
33         {
34             int searchKey;
35             try
36             {
37                 searchKey = Int32.Parse(tbKey.Text);
38                 lblResultSet.Text = "";
39
40                 // perform the binary search
41                 int element = BinarySearch(a, searchKey, 0, a.Length - 1);
42
43                 if (element != -1)
44                     lblExplanation.Text = "Found value at element " + element;
45                 else
46                     lblExplanation.Text = "Value not found";
47             }
48             catch (Exception ex)
49             {
50                 lblResultSet.Text = ex.Message.ToUpperInvariant();
51             }
52         }
53
54         /// <summary>
55         ///     Binary Search method to find given key in given array that offered by book.
56         /// </summary>
57         /// <param name="array">array which will be searched to find keys</param>
58         /// <param name="key">element which will be searched in array</param>
59         /// <returns></returns>
60         0 references
61         public int BinarySearch(int[] array, int key)
62         {
63             int low = 0; // low subscript
64             int high = array.Length - 1; // high subscript
65             int middle; // middle subscript
66
67             while (low <= high)
68             {
69                 middle = (low + high) / 2;
70
71                 // the following line displays the portion
72                 // of the array currently being manipulated during
73                 // each iteration of the binary search loop
74                 BuildOutput(a, low, middle, high);
75
76                 if (key == array[middle]) // match
77                     return middle;
78                 else if (key < array[middle])
79                     high = middle - 1; // search low end of array
80                 else
81                     low = middle + 1;
82             } // end BinarySearch
83
84             return -1; // search key not found
85         }
86
87         /// <summary>
88         ///     Binary Search method to find given key in given array recursively.
89         /// </summary>
90         /// <param name="array">array which will be searched to find keys</param>
91         /// <param name="key">element which will be searched in array</param>
92         /// <param name="start">first index to start searching in array</param>
93         /// <param name="end">last index to end searching in arrays</param>
94         /// <returns></returns>
95         3 references
96         public int BinarySearch(int[] array, int key, int start, int end)
97         {

```

```

98     if (end >= start)
99     {
100         int mid = start + (end - start) / 2;
101
102         // If the element is present at the
103         // middle itself
104         if (array[mid] == key)
105         {
106             BuildOutput(a, start, mid, end);
107             return mid;
108         }
109
110
111         // If element is smaller than mid, then
112         // it can only be present in left subarray
113         if (array[mid] > key)
114         {
115             BuildOutput(a, start, mid, end);
116             return BinarySearch(array, key, start, mid - 1);
117         }
118
119
120         // Else the element can only be present
121         // in right subarray
122         BuildOutput(a, start, mid, end);
123         return BinarySearch(array, key, mid + 1, end);
124     }
125
126     return -1; // search key not found
127
128 }
129

```

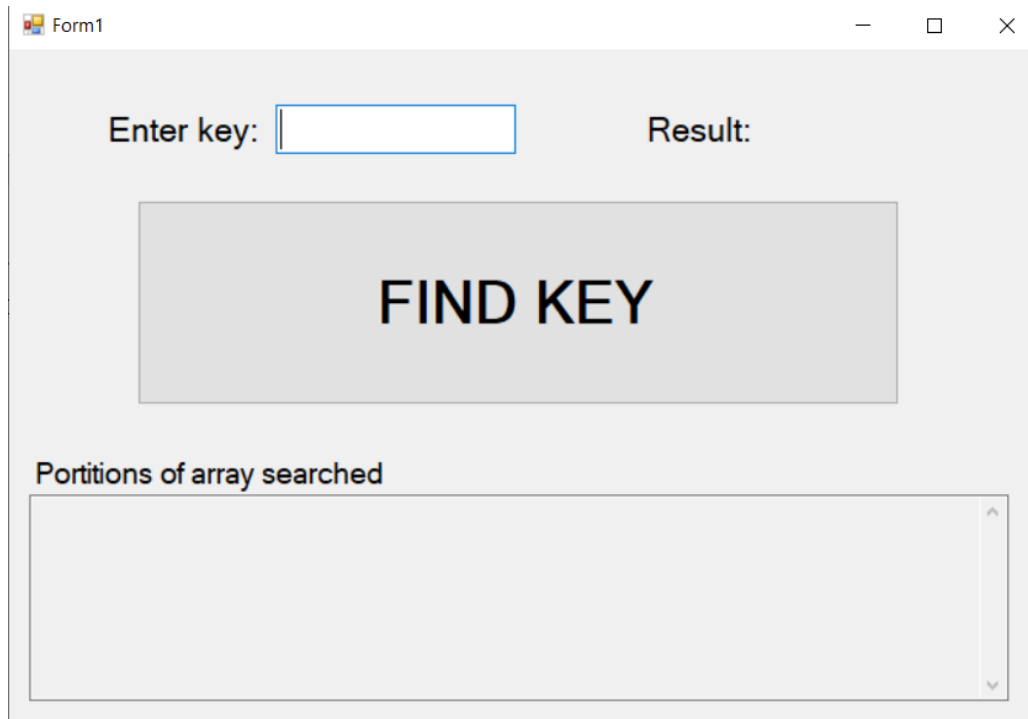
```

130
131 /// <summary>
132 ///     This method prints the search steps in lblResultSet tool.
133 /// </summary>
134 /// <param name="array">array which is searched to find key</param>
135 /// <param name="low">low subscript</param>
136 /// <param name="mid">middle subscript</param>
137 /// <param name="high">high subscript</param>
138 4 references
139 public void BuildOutput(int[] array, int low, int mid, int high)
140 {
141     for (int i = 0; i < array.Length; i++)
142     {
143         if (i < low || i > high)
144             lblResultSet.Text += " ";
145
146         // mark middle element in output
147         else if (i == mid)
148             lblResultSet.Text += array[i].ToString("00") + "* ";
149         else
150             lblResultSet.Text += array[i].ToString("00") + " ";
151     }
152
153     lblResultSet.Text += "\r\n";
154 } // end BuildOutput
155
156
157
158
159

```

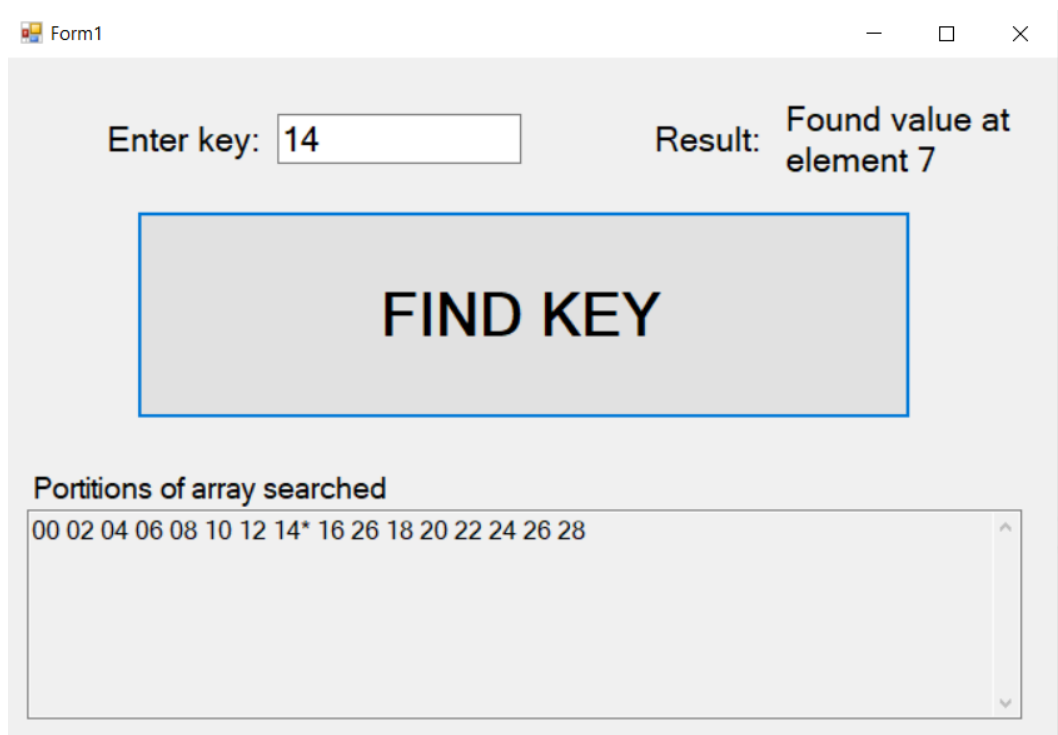
Outputs of program:

- Starting the application:



The screenshot shows a Windows application window titled 'Form1'. It contains a text input field labeled 'Enter key:' which is empty. To its right is the label 'Result:'. Below these is a large gray button with the text 'FIND KEY'. At the bottom, there is a label 'Portions of array searched' above a text area that is currently empty.

- Binary Search examples:



The screenshot shows the same 'Form1' application window. The 'Enter key:' text box now contains the value '14'. The 'Result:' label now displays 'Found value at element 7'. The 'FIND KEY' button remains visible. The 'Portions of array searched' text area now displays the sequence of indices: '00 02 04 06 08 10 12 14\* 16 26 18 20 22 24 26 28', where the asterisk is positioned below the '14'.

Form1

Enter key:  Result: Found value at element 0

**FIND KEY**

Portions 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:  Result: Value not found

**FIND KEY**

Portions 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:

Form1

Enter key:

Result: Value not found

**FIND KEY**

Portions of array searched

INPUT STRING WAS NOT IN A CORRECT FORMAT.

- Responsive design:

Form1

Enter key:

Result: Found value at element 6

**FIND KEY**

Portions 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\*

Form1

Enter key:

Result: Found value at element 6

FIND KEY

Portions 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\*