GITAM

Deemed to be University

(Estd. u/s of UGC Act, 1950)

Hyderabad Campus

Laboratory Record Book

Name …KARRI SRI SAI SOWMYA……..

Department of …CSE……. Reg. No. ...221710310031..

Laboratory ……IDE…..….. Section …B-10…..

GITAM

Deemed to be University

(Estd. u/s of UGC Act, 1950)

Hyderabad Campus

CERTIFICATE

Certified that this is the bonafide record of practical work done by Mr./Ms. …… with Reg. No. …….. of B.Tech..... branch in …… Laboratory of Department of …….. during the academic year ……..

Faculty I/c

Date : ***Head of the Department***

INDEX

WEEK – 1 (26 AUG 2020)

INTRODUCTION TO IDE

WHAT IS IDE?

- An integrated development environment is a software application that provides comprehensive facilities to computer programmers for software development.
- An IDE normally consists of at least a source code editor, build automation tools and a debugger.

KEY BENEFITS

- Code completion capabilities improve programming workflow.
- Automatically checks for errors to ensure top quality code.
- Refactoring capabilities allow developers to make comprehensive and mistake-free renaming changes.
- Maintain a smooth development cycle.
- Increase developer efficiency and satisfaction.
- Deliver top-quality software on schedule.

IDE FEATURES

- **Text Editors:**

  Virtually every IDE will have a text editor designed to write and manipulate source code. Some tools may have visual components to drag and drop front-end components, but most have a simple interface with language-specific syntax highlighting

- **Debugger:**

  Debugging tools assist users in identifying and remedying errors within source code. They often simulate real-world scenarios to test functionality and performance. Programmers and software engineers can usually test the various segments of code and identify errors before the application is released.

- **Compiler:**

  Compilers are components that translate programming language into a form machines can process, such as binary code. The machine code is analyzed to ensure its accuracy. The compiler then parses and optimizes the code to optimize performance.

- **Code Completion:**

  Code complete features assist programmers by intelligently identifying and inserting common code components. These features save developers time writing code and reduce the likelihood of typos and bugs.

- **Programming language support:**

  IDEs are typically specific to a single programming language, though several also offer multi-language support. As such, the first step is to figure out which languages you will be coding in and narrow your prospective IDE list down accordingly. Examples include Ruby, Python, and Java IDE tools.

- **Integrations and plugins:**

  With the name integrated development environment, it is no surprise that integrations need to be considered when looking at IDEs. Your IDE is your development portal, so being able to incorporate all your other development tools will improve development workflows and productivity. Poor integrations can cause numerous issues and lead to many headaches, so make sure you understand how well a potential IDE fits into your ecosystem of existing tools.

KEY TERMS:

- **Compiler:** Converts source code to object code.
- **Debugging**: The process of removing errors from a program. 1) compiler 2) linker 3) logic
- **Linker**: Connects or links object files into an executable file.
- **Loader:** Part of the operating system that loads executable files into memory and directs the CPU to start running the program.
- **Pre-Processor:** The first step the compiler does in converting source code to object code.
- **Text Editor:** A software program for creating and editing ASCII text files.
- **Warning:** A compiler alert that there might be a problem.

APPLICATIONS

- Console
- Windows
- Client/Server
- Web based
- Enterprise applications
- SharePoint servers

EXAMPLES:

LANGUAGE	IDE NAMES
C/C++	Code::Blocks, Bloodshed Dev-C++
Java	Eclipse, NetBeans
.NET	Visual Studio
Python	PyCharm ,Komodo IDE,AWS Cloud 9

CONSOLE APPLICATION – USING C#.NET

INTRO:

Purpose

Install .NET and create your first application.

Prerequisites

None.

Time to Complete

10 minutes

Scenario

A simple application written in C# that prints Hello, World! to the console.

Let's get started

Click on **LETS GET STARTED**

DOWNLOAD AND INSTALL

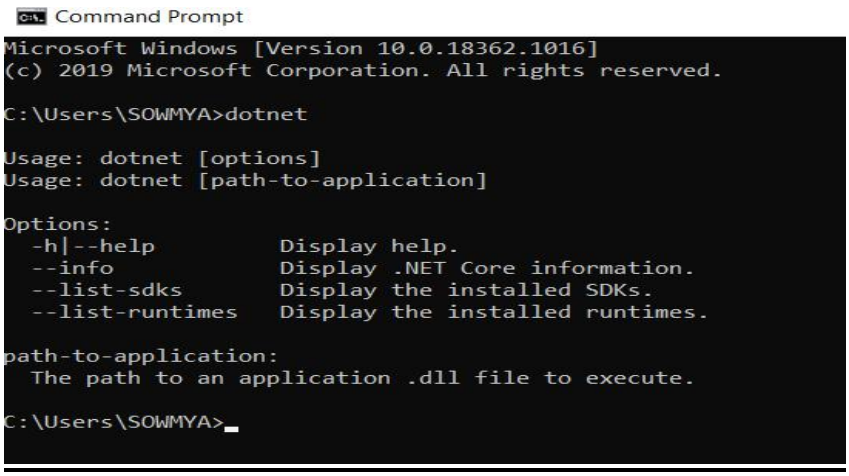
To start building .NET apps, download and install the .NET SDK (Software Development Kit).

Download .NET SDK (64-bit)

Click on **Download .NET SDK**

Check Everything Is Installed:

Once you've installed, open a **new** command prompt and run the following command:



The command runs, printing out information about how to use dotnet.

CREATE YOUR APP

In your command prompt, run the following commands:

```
C:\Users\SOWMYA>dotnet new console -o myApp
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on myApp\myApp.csproj...
  Determining projects to restore...
  Restored C:\Users\SOWMYA\myApp\myApp.csproj (in 240 ms).

Restore succeeded.


C:\Users\SOWMYA>cd myApp

C:\Users\SOWMYA\myApp>
```

The dotnet command creates a new application of type console for you. The -o parameter creates a directory named myApp where your app is stored, and populates it with the required files. The cd myApp command puts you into the newly created app directory.

The main file in the myApp folder is Program.cs. By default, it already contains the necessary code to write "Hello World!" to the Console.

*Program - Notepad

File Edit Format View Help

```
using System;

namespace myApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

RUN YOUR APP

In your command prompt, run the following command:

```
C:\Users\SOWMYA\myApp>dotnet run
Hello World!

C:\Users\SOWMYA\myApp>
```

We've built and run your first .NET app!

EDIT YOUR APP

Open Program.cs in any text editor, such as Notepad, and add a new line of code below the one that prints "Hello World!", like the following:

*Program - Notepad

File Edit Format View Help

```
using System;

namespace myApp
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
            Console.WriteLine("The current time is " + DateTime.Now);
        }
    }
}
```

Save the Program.cs file, and run your code again

```
C:\Users\SOWMYA\myApp>dotnet run
Hello World!
The current time is 28-08-2020 22:40:33

C:\Users\SOWMYA\myApp>
```

NEXT,

Click on **Try .NET in your browser**

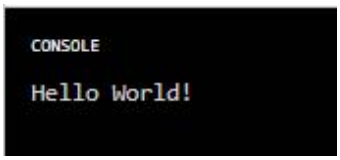
.NET In-browser tutorial gets opened.

Step 1: Intro

This is a .NET Console application written in C#.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace myApp
{
   class Program
   {
      static void Main()
      {
         Console.WriteLine("Hello World!");
      }
   }
}
```

Output:

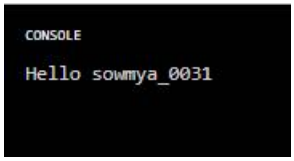


Step 2: Strings

Try modifying the code so that the console says hello to your name, instead of the world .

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace myApp
{
   class Program
   {
      static void Main()
      {
         Console.WriteLine("Hello sowmya_0031");
      }
   }
}
```

Output:



Step 3: Variables

Variables hold values that you can use elsewhere in your code.

Let's store your name in a variable, then read the value from that variable when creating the output message.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace myApp
{
   class Program
   {
      static void Main()
      {
         var name = "sowmya_0031";
         Console.WriteLine("Hello " + name + "!");
      }
   }
}
```

Output :

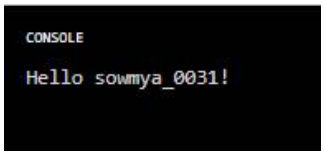
Step 4: String interpolation

String interpolation lets you piece together strings in a more concise and readable way.

If you add a $ before the opening quotes of the string, you can then include string values, like the name variable, inside the string in curly brackets. Try it out and select **Run Code**.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace myApp
{
   class Program
   {
      static void Main()
      {
         var name = "sowmya_0031";
         Console.WriteLine($"Hello {name}!");
      }
   }
}
```

Output:



Step 5: Methods

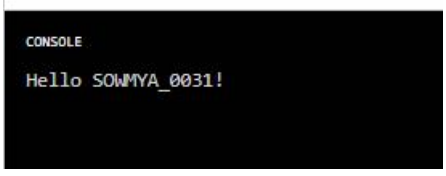
Methods take inputs, do some work, and sometimes return a result.

ToUpper() is a method you can invoke on a string, like the name variable. It will return the same string, converted to uppercase.

Update the greeting to change the name of the person being greeted to uppercase, and select **Run Code**.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace myApp
{
   class Program
   {
      static void Main()
      {
         var name = "sowmya_0031";
         Console.WriteLine($"Hello {name.ToUpper()}!");
      }
   }
}
```

Output:



Step 6: Collections

Collections hold multiple values of the same type.

Replace the **name** variable with a **names** variable that has a list of names. Then use a **foreach** loop to iterate over all the names and say hello to each person.

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace myApp
{
   class Program
   {
      static void Main()
      {
         var names = new List<string> { "sowmya_0031", "Felipe", "Emillia" };
         foreach (var name in names)
         {
            Console.WriteLine($"Hello {name.ToUpper()}!");
         }
      }
   }
}
```

Output:

NEXT,
Click on **Tutorial: numbers** and **Begin**

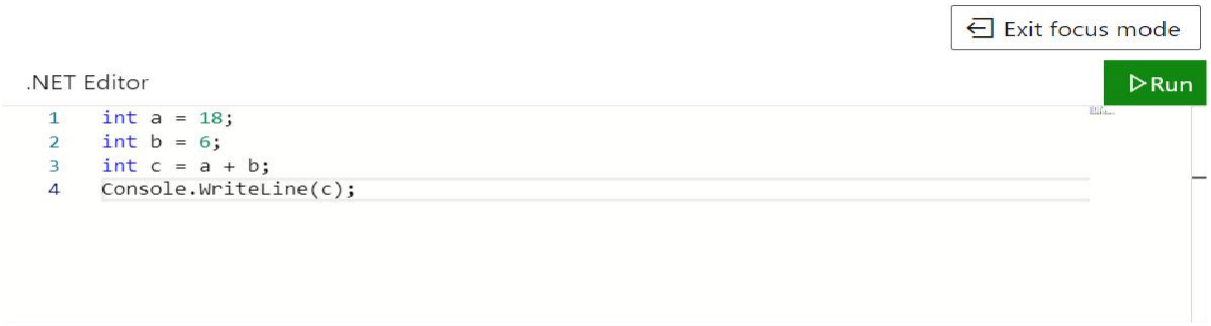
NUMBERS IN C# :

MANIPULATE INTEGRAL AND FLOATING POINT NUMBERS IN C#**:**

Explore Integer Math:

Run the following code in the interactive window. Select the **Enter focus mode** button. Then, type the following code block in the interactive window and select **Run**:

```
int a = 18;
int b = 6;
int c = a + b;
Console.WriteLine(c);
```

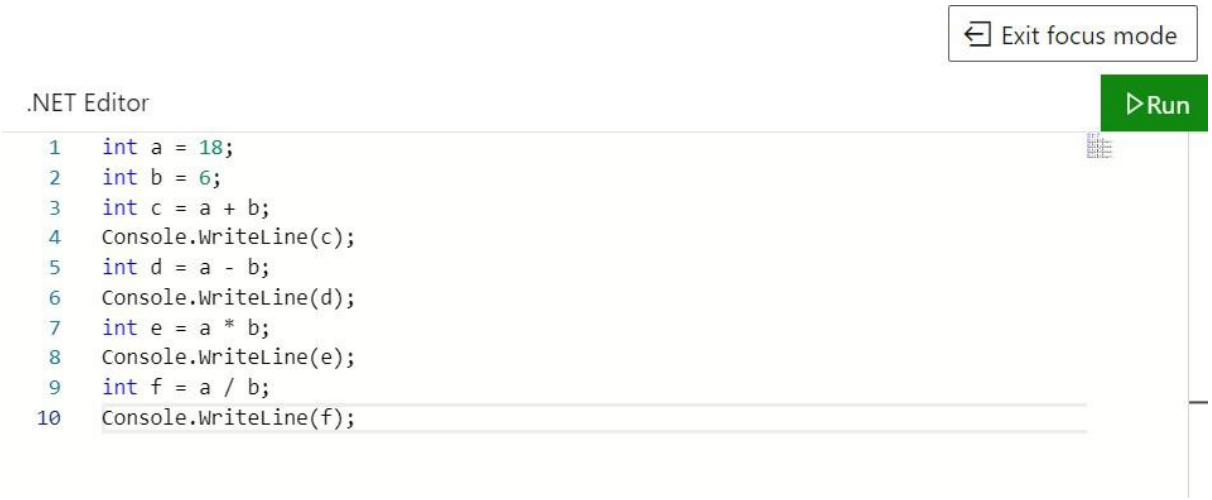


Click on **Run ,** we'll get the below output

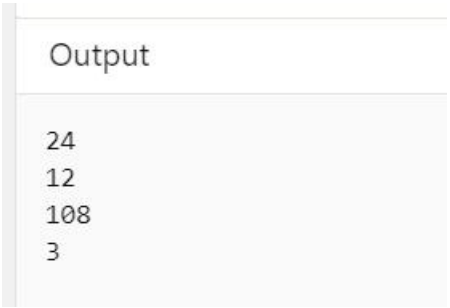
We've seen one of the fundamental math operations with integers. The int type represents an **integer**, a positive or negative whole number. We use the + symbol for addition. Other common mathematical operations for integers include:

- - for subtraction
- * for multiplication
- / for division

Start by exploring those different operations. Modify the third line to try each of these operations. After each edit.



select the **Run** button



WEEK – 2 (2 SEPT 2020)

NUMBERS IN C# :

Explore Order Of Operations:

The C# language defines the precedence of different mathematics operations with rules consistent with the rules you learned in mathematics. Multiplication and division take precedence over addition and subtraction. We are Exploring that by running the following code in the interactive window:

```
int a = 6;
int b = 5;
int c = 3;
int d = a + b * c;
Console.WriteLine(d);
```

Output:

```
Output
21
```

The output demonstrates that the multiplication is performed before the addition. We can force a different order of operation by adding parentheses around the operation or operations you want performed first:

```
int a = 5;
int b = 4;
int c = 2;
int d = (a + b) * c;
Console.WriteLine(d);
```

Output:

```
Output
18
```

Replace the fourth line above with different operations,

```
int a = 5;
int b = 4;
int c = 2;
int d = (a + b) - 6 * c + (12 * 4) / 3 + 12;
Console.WriteLine(d);
```

Output :

```
Output

25
```

Integer division always produces an integer result, even when you'd expect the result to include a decimal or fractional portion. We can see this by running the below code :

```
int a = 7;
int b = 11;
int c = 3;
int d = (a + b) / c;
Console.WriteLine(d);
```

Output:

```
Output

6
```

Explore Integer Precision and Limits:

That last sample showed that integer division truncates the result. we can get the **remainder** by using the **remainder** operator, the % character:

```
int a = 7;
int b = 4;
int c = 3;
int d = (a + b) / c;
int e = (a + b) % c;
Console.WriteLine($"quotient: {d}");
Console.WriteLine($"remainder: {e}");
```

Output:

```
Output

quotient: 3
remainder: 2
```

The C# integer type differs from mathematical integers in one other way: the int type has minimum and maximum limits. Run this code to see those limits:

```
int max = int.MaxValue;
int min = int.MinValue;
Console.WriteLine($"The range of integers is {min} to {max}");
```

Output:

```
Output

The range of integers is -2147483648 to 2147483647
```

If a calculation produces a value that exceeds those limits, you have an **underflow** or **overflow** condition. The answer appears to wrap from one limit to the other.

```
int max = int.MaxValue;
int min = int.MinValue;
Console.WriteLine($"The range of integers is {min} to {max}");
int what = max + 3;
Console.WriteLine($"An example of overflow: {what}");
```

Output:

```
Output

The range of integers is -2147483648 to 2147483647
An example of overflow: -2147483646
```

The answer is very close to the minimum (negative) integer. It's the same as min + 2. The addition operation **overflowed** the allowed values for integers. The answer is a very large negative number because an overflow "wraps around" from the largest possible integer value to the smallest.

UNDERFLOW:

```
int max = int.MaxValue;
int min = int.MinValue;
int what = min - 3;
Console.WriteLine($"An example of underflow: {what}");
```

Output:

```
Output

An example of underflow: 2147483645
```

Work With The Double Type:

There are other numeric types with different limits and precision that you would use when the int type doesn't meet your needs.
the double numeric type represents a double-precision floating point number. Those terms may be new to you. A **floating point** number is useful to represent non-integral numbers that may be very large or small in magnitude. **Double-precision** is a relative term that describes the numbers of binary digits used to store the value. **Double precision** number have twice the number of binary digits as **single-precision**. On modern computers, it is more common to use double precision than single precision numbers. **Single precision** numbers are declared using the float keyword.
Let's run the following code

```
double a = 5;
double b = 4;
double c = 2;
double d = (a + b) / c;
Console.WriteLine(d);
```

Output:

```
Output
4.5
```

slightly more complicated expression with doubles:

```
double a = 19;
double b = 23;
double c = 8;
double d = (a + b) / c;
Console.WriteLine(d);
```

Output:

```
Output
5.25
```

The range of a double value is much greater than integer values.

```
double max = double.MaxValue;
double min = double.MinValue;
Console.WriteLine($"The range of double is {min} to {max}");
```

Output:

```
Output

The range of double is -1.79769313486232E+308 to 1.79769313486232E+308
```

These values are printed out in scientific notation. The number to the left of the E is the significand. The number to the right is the exponent, as a power of 10.

Just like decimal numbers in math, doubles in C# can have rounding error.

```
double third = 1.0 / 3.0;
Console.WriteLine(third);
```

Output:

```
Output

0.333333333333333
```

0.3 is 3/10 and not exactly the same as 1/3. Similarly, 0.33 is 33/100. That's closer to 1/3, but still not exact.

Other calculations with large numbers, small numbers, multiplication, and division using the double type. Try more complicated calculations:

```
double a = 25;
double b = 14;
double c = 6;
double d = (a + c) / b;
Console.WriteLine(d);
```

```
Output

2.21428571428571
```

```
double a = 15;
double b = 9;
double c = 12;
double d = (a * b) / c;
Console.WriteLine(d);
```

```
Output

11.25
```

```
double a = 5;
double b = 4;
double c = 2;
double d = (a + b) - b * c + (a * c )/b;
Console.WriteLine(d);
```

```
Output

3.5
```

Work With Decimal Types:

There's one other type to learn: the decimal type. The decimal type has a smaller range but greater precision than double. Let's take a look:

```
decimal min = decimal.MinValue;
decimal max = decimal.MaxValue;
Console.WriteLine($"The range of the decimal type is {min} to {max}");
```

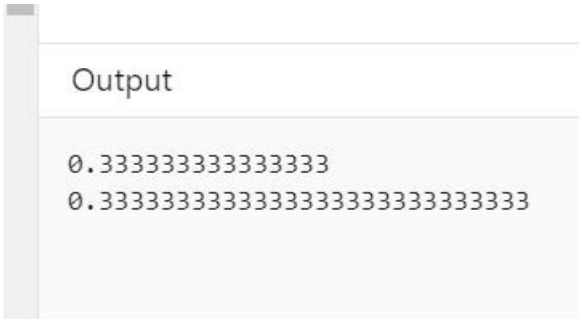
Output:

```
Output
The range of the decimal type is -79228162514264337593543950335 to 79228162514264337593543950335
```

the range is smaller than the double type. we can see the greater precision with the decimal type by trying the following code:

```
double a = 1.0;
double b = 3.0;
Console.WriteLine(a / b);
decimal c = 1.0M;
decimal d = 3.0M;
Console.WriteLine(c / d);
```

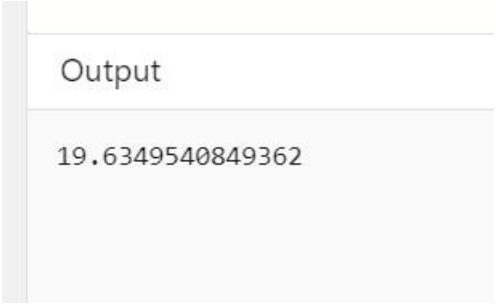
Output:



The math using the decimal type has more digits to the right of the decimal point.

The M suffix on the numbers is how we indicate that a constant should use the decimal type. Otherwise, the compiler assumes the double type

```
double radius = 2.50;
double area = Math.PI * radius * radius;
Console.WriteLine(area);
```



code that calculates the area of a circle whose radius is 2.50 centimeters. The area of a circle is the radius squared multiplied by PI. .NET contains a constant for PI, Math.PI that We can use for that value. Math.PI, like all constants declared in the System.Math namespace, is a double value. For that reason, you should use double instead of decimal values .

The "Numbers in C#" is completed,

WEEK 3 (09-09-2020)

BRANCHES AND LOOPS IN C#:

To write code that examines variables and changes execution path based on those variables. Our browser to write C# is used interactively and the results of compiling and running our code is seen . we are going to perform a series of lessons that explore branching and looping constructs in C#. These lessons teach us the fundamentals of the C# language.

Make Decisions Using The If Statements:

Run the following code:

```
int a = 5;
int b = 6;
if (a + b > 10)
    Console.WriteLine("The answer is greater than 10.");
```

Output:

```
Output

The answer is greater than 10.
```

Modify the declaration of b so that the sum is less than 10:

```
int a = 5;
int b = 3;
if (a + b > 10)
    Console.WriteLine("The answer is greater than 10");
else
    Console.WriteLine("The answer is not greater than 10");
```

Output:

```
Output

The answer is not greater than 10
```

Select the **Run** button again. Because the answer is less than 10, nothing is printed.
The **condition** we're testing is false. we don't have any code to execute because we've only written one of the possible branches for an if statement: the true branch.

```
int a = 5;
int b = 3;
if (a + b > 10)
{
    Console.WriteLine("The answer is greater than 10");
}
else
{
    Console.WriteLine("The answer is not greater than 10");
}
```

Output:

```
Output

The answer is not greater than 10
```

This first sample shows the power of if and boolean types. A *boolean* is a variable that can have one of two values: true or false. C# defines a special type, bool for Boolean variables. The if statement checks the value of a bool. When the value is true, the statement following the if executes. Otherwise, it's skipped.

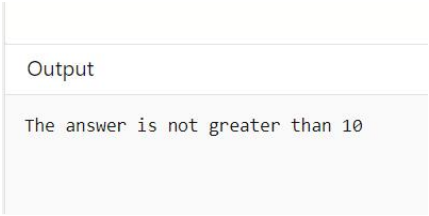
This process of checking conditions and executing statements based on those conditions is powerful.

Make If Else Work Together :

To execute different code in both the true and false branches,we create an else branch that executes when the condition is false.

```
int a = 5;
int b = 3;
if (a + b > 10)
    Console.WriteLine("The answer is greater than 10");
else

    Console.WriteLine("The answer is not greater than 10");
```

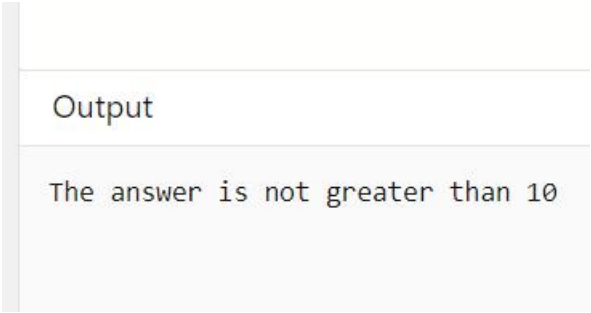
Output:



The statement following the else keyword executes only when the condition being tested is false. Combining if and else with boolean conditions provides all the power you need. Because indentation isn't significant, we need to use { and } to indicate when you want more than one statement to be part of the block that executes conditionally. C# programmers typically use those braces on all if and else clauses.

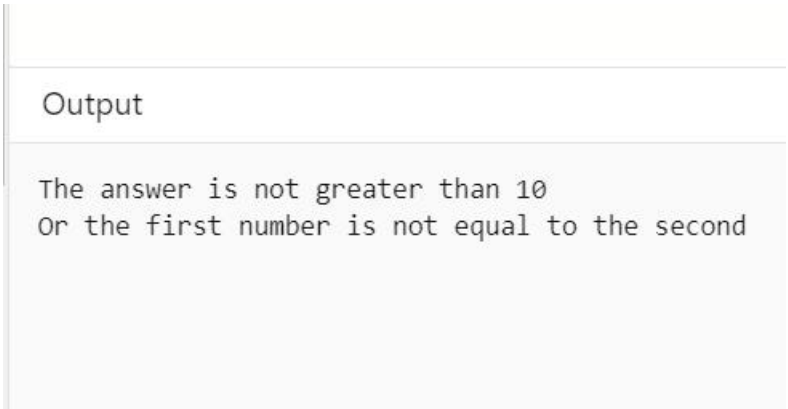
```
int a = 5;
int b = 3;
if (a + b > 10)
{
    Console.WriteLine("The answer is greater than 10");
}
else
{
    Console.WriteLine("The answer is not greater than 10");
}
```

Output:




```
int a = 5;
int b = 3;
int c = 4;
if ((a + b + c > 10) && (a == b))
{
    Console.WriteLine("The answer is greater than 10");
    Console.WriteLine("And the first number is equal to the second");
}
else
{
    Console.WriteLine("The answer is not greater than 10");
    Console.WriteLine("Or the first number is not equal to the second");
}
```

Output:



The == symbol tests for *equality*. Using == distinguishes the test for equality from assignment, which you saw in a = 5.

The && represents "and". It means both conditions must be true to execute the statement in the true branch. These examples also show that we can have multiple statements in each conditional branch, provided you enclose them in { and }.

we can also use || to represent "or":

```
int a = 5;
int b = 3;
int c = 4;
if ((a + b + c > 10) || (a == b))
{
    Console.WriteLine("The answer is greater than 10");
    Console.WriteLine("Or the first number is equal to the second");
}
else
{
    Console.WriteLine("The answer is not greater than 10");
    Console.WriteLine("And the first number is not equal to the second");
}
```

Output:

```
Output

The answer is greater than 10
Or the first number is equal to the second
```

We can Modify the values of a, b, and c and switch between && and || to explore

Use Loops To Repeat Operations:

Another important concept to create larger programs is **loops**. You'll use loops to repeat statements that you want executed more than once.

```
int counter = 0;
while (counter < 10)
{
  Console.WriteLine($"Hello World! The counter is {counter}");
  counter++;
}
```

Output:

```
Hello World! The counter is 0
Hello World! The counter is 1
Hello World! The counter is 2
Hello World! The counter is 3
Hello World! The counter is 4
Hello World! The counter is 5
Hello World! The counter is 6
Hello World! The counter is 7
Hello World! The counter is 8
Hello World! The counter is 9
```

The while statement checks a condition and executes the statement following the while. It will repeat checking the condition and executing those statements until the condition is false.

There's one other new operator. The ++ after the counter variable is the **increment** operator. It adds 1 to the value of counter, and stores that value in the counter variable.

The while loop tests the condition before executing the code following the while.
The do ... while loop executes the code first, and then checks the condition. It looks like this:

```
int counter = 0;
do
{
  Console.WriteLine($"Hello World! The counter is {counter}");
  counter++;
} while (counter < 10);
```

Output:

```
Hello World! The counter is 0
Hello World! The counter is 1
Hello World! The counter is 2
Hello World! The counter is 3
Hello World! The counter is 4
Hello World! The counter is 5
Hello World! The counter is 6
Hello World! The counter is 7
Hello World! The counter is 8
Hello World! The counter is 9
```

```
for(int counter = 0; counter < 10; counter++)
{
  Console.WriteLine($"Hello World! The counter is {counter}");
}
```

Output:

```
Hello World! The counter is 0
Hello World! The counter is 1
Hello World! The counter is 2
Hello World! The counter is 3
Hello World! The counter is 4
Hello World! The counter is 5
Hello World! The counter is 6
Hello World! The counter is 7
Hello World! The counter is 8
Hello World! The counter is 9
```

This do loop and the earlier while loop work the same.

```
for (int row = 1; row < 11; row++)
{
  Console.WriteLine($"The row is {row}");
}
```

Output:

```
The row is 1
The row is 2
The row is 3
The row is 4
The row is 5
The row is 6
The row is 7
The row is 8
The row is 9
The row is 10
```

Work With The For Loop:

Another common loop statement that we'll see in C# code is the for loop.

```
for (char column = 'a'; column < 'k'; column++)
{
  Console.WriteLine($"The column is {column}");
}
```

Output:

```
The column is a
The column is b
The column is c
The column is d
The column is e
The column is f
The column is g
The column is h
The column is i
The column is j
```

This does the same work as the while loop and the do loop we've already used.
The for statement has three parts that control how it works.

The first part is the **for initializer**: int counter = 0; declares that counter is the loop variable, and sets its initial value to 0.

The middle part is the **for condition**: counter < 10 declares that this for loop continues to execute as long as the value of counter is less than 10.

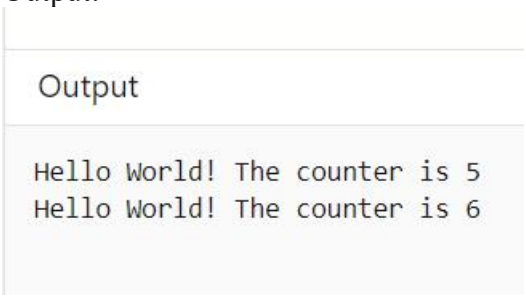
The final part is the **for iterator**: counter++ specifies how to modify the loop variable after executing the block following the for statement. Here, it specifies that counter should be incremented by 1 each time the block executes.

We can experiment the following:

- Change the initializer to start at a different value.
- Change the condition to stop at a different value.

```
for(int counter = 5; counter < 7; counter++)
  {
    Console.WriteLine($"Hello World! The counter is {counter}");
  }
```

Output:



There's one other looping statement the foreach statement. The foreach statement repeats its statement for every item in a sequence of items. It's most often used with *collections*, so it is covered in the next tutorial.

Created Nested Loops :

A while, do, or for loop can be nested inside another loop to create a matrix using the combination of each item in the outer loop with each item in the inner loop.

To build a set of alphanumeric pairs to represent rows and columns:

One for loop can generate the rows:

```
for (int row = 1; row < 11; row++)
{
  Console.WriteLine($"The row is {row}");
}
```

Output:

```
The row is 1
The row is 2
The row is 3
The row is 4
The row is 5
The row is 6
The row is 7
The row is 8
The row is 9
The row is 10
```

Another loop can generate the columns:

```
for (char column = 'a'; column < 'k'; column++)
{
  Console.WriteLine($"The column is {column}");
}
```

```
Output:
The column is a
The column is b
The column is c
The column is d
The column is e
The column is f
The column is g
The column is h
```

```
The column is i
The column is j
```

we can nest one loop inside the other to form pairs:

```
for (int row = 1; row < 11; row++)
{
  for (char column = 'a'; column < 'k'; column++)
  {
    Console.WriteLine($"The cell is ({row}, {column})");
  }
}
```

Output:

```
The cell is (1, a)
The cell is (1, b)
The cell is (1, c)
The cell is (1, d)
The cell is (1, e)
The cell is (1, f)
The cell is (1, g)
The cell is (1, h)
The cell is (1, i)
The cell is (1, j)
The cell is (2, a)
The cell is (2, b)
The cell is (2, c)
The cell is (2, d)
The cell is (2, e)
The cell is (2, f)
The cell is (2, g)
The cell is (2, h)
The cell is (2, i)
The cell is (2, j)
The cell is (3, a)
The cell is (3, b)
The cell is (3, c)
The cell is (3, d)
The cell is (3, e)
The cell is (3, f)
The cell is (3, g)
The cell is (3, h)
The cell is (3, i)
The cell is (3, j)
The cell is (4, a)
```

```
The cell is (4, b)
The cell is (4, c)
The cell is (4, d)
The cell is (4, e)
The cell is (4, f)
The cell is (4, g)
The cell is (4, h)
The cell is (4, i)
The cell is (4, j)
The cell is (5, a)
The cell is (5, b)
The cell is (5, c)
The cell is (5, d)
The cell is (5, e)
The cell is (5, f)
The cell is (5, g)
The cell is (5, h)
The cell is (5, i)
The cell is (5, j)
The cell is (6, a)
The cell is (6, b)
The cell is (6, c)
The cell is (6, d)
The cell is (6, e)
The cell is (6, f)
The cell is (6, g)
The cell is (6, h)
The cell is (6, i)
The cell is (6, j)
The cell is (7, a)
The cell is (7, b)
The cell is (7, c)
The cell is (7, d)
The cell is (7, e)
The cell is (7, f)
The cell is (7, g)
The cell is (7, h)
The cell is (7, i)
The cell is (7, j)
The cell is (8, a)
The cell is (8, b)
The cell is (8, c)
The cell is (8, d)
The cell is (8, e)
The cell is (8, f)
The cell is (8, g)
The cell is (8, h)
The cell is (8, i)
```

```
The cell is (8, j)
The cell is (9, a)
The cell is (9, b)
The cell is (9, c)
The cell is (9, d)
The cell is (9, e)
The cell is (9, f)
The cell is (9, g)
The cell is (9, h)
The cell is (9, i)
The cell is (9, j)
The cell is (10, a)
The cell is (10, b)
The cell is (10, c)
The cell is (10, d)
The cell is (10, e)
The cell is (10, f)
The cell is (10, g)
The cell is (10, h)
The cell is (10, i)
The cell is (10, j)
```

We can see that the outer loop increments once for each full run of the inner loop.

Reversing the row and column nesting:

```
for (char column = 'a'; column < 'k'; column++)
{
    for (int row = 1; row < 11; row++)
  {
  Console.WriteLine($"The cell is ({row}, {column})");
  }
}
```

Output:

```
The cell is (1, a)
The cell is (2, a)
The cell is (3, a)
The cell is (4, a)
The cell is (5, a)
The cell is (6, a)
The cell is (7, a)
The cell is (8, a)
The cell is (9, a)
```

```
The cell is (10, a)
The cell is (1, b)
The cell is (2, b)
The cell is (3, b)
The cell is (4, b)
The cell is (5, b)
The cell is (6, b)
The cell is (7, b)
The cell is (8, b)
The cell is (9, b)
The cell is (10, b)
The cell is (1, c)
The cell is (2, c)
The cell is (3, c)
The cell is (4, c)
The cell is (5, c)
The cell is (6, c)
The cell is (7, c)
The cell is (8, c)
The cell is (9, c)
The cell is (10, c)
The cell is (1, d)
The cell is (2, d)
The cell is (3, d)
The cell is (4, d)
The cell is (5, d)
The cell is (6, d)
The cell is (7, d)
The cell is (8, d)
The cell is (9, d)
The cell is (10, d)
The cell is (1, e)
The cell is (2, e)
The cell is (3, e)
The cell is (4, e)
The cell is (5, e)
The cell is (6, e)
The cell is (7, e)
The cell is (8, e)
The cell is (9, e)
The cell is (10, e)
The cell is (1, f)
The cell is (2, f)
The cell is (3, f)
The cell is (4, f)
The cell is (5, f)
The cell is (6, f)
The cell is (7, f)
```

```
The cell is (8, f)
The cell is (9, f)
The cell is (10, f)
The cell is (1, g)
The cell is (2, g)
The cell is (3, g)
The cell is (4, g)
The cell is (5, g)
The cell is (6, g)
The cell is (7, g)
The cell is (8, g)
The cell is (9, g)
The cell is (10, g)
The cell is (1, h)
The cell is (2, h)
The cell is (3, h)
The cell is (4, h)
The cell is (5, h)
The cell is (6, h)
The cell is (7, h)
The cell is (8, h)
The cell is (9, h)
The cell is (10, h)
The cell is (1, i)
The cell is (2, i)
The cell is (3, i)
The cell is (4, i)
The cell is (5, i)
The cell is (6, i)
The cell is (7, i)
The cell is (8, i)
The cell is (9, i)
The cell is (10, i)
The cell is (1, j)
The cell is (2, j)
The cell is (3, j)
The cell is (4, j)
The cell is (5, j)
The cell is (6, j)
The cell is (7, j)
The cell is (8, j)
The cell is (9, j)
The cell is (10, j)
```

Combine Branches And Loops :

Now that we've seen the if statement and the looping constructs in the C# language,

C# code to find the sum of all integers 1 through 20 that are divisible by 3.

```
int sum = 0;
for (int number = 1; number < 21; number++)
{
  if (number % 3 == 0)
  {
    sum = sum + number;
  }
}
Console.WriteLine($"The sum is {sum}");
```

Output:

```
Output

The sum is 63
```

The % operator gives you the remainder of a division operation.

- The if statement gives you the condition to see if a number should be part of the sum.
- The for loop can help you repeat a series of steps for all the numbers 1 through 20.

"branches and loops" in C# completed

THE LIST COLLECTIONS :

It contains a series of lessons that create, modify, and explore collections and arrays.

```
var names = new List<string> { "<Sowmya_0031>", "Ana", "Felipe" };
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
```

Output:

We've created a list of strings, added three names to that list, and printed out the names in all CAPS. we're using concepts that we've learned in earlier tutorials to loop through the list.

The code to display names makes use of the string interpolation feature. When you precede a string with the $ character, we can embed C# code in the string declaration. The actual string replaces that C# code with the value it generates. here, it replaces the {name.ToUpper()} with each name, converted to capital letters, because you called the String.ToUpper method.

```
var names = new List<string> { "<Sowmya_0031>", "Ana", "Felipe" };
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
Console.WriteLine();
names.Add("Maggie");
names.Add("Navya");
names.Remove("Sowmya_0031");
```

```
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
```

Output:

```
Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!

Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!
Hello MAGGIE!
Hello NAVYA!
```

We've added two more names to the end of the list. We've also removed one as well. The output from this block of code shows the initial contents, then prints a blank line and the new contents.

The List<T> enables you to reference individual items by **index** as well. You access items using the [and] tokens. Add the following code below what you've already written and try it

```
var names = new List<string> { "<Sowmya_0031>", "Ana", "Felipe" };
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
Console.WriteLine();
names.Add("Maggie");
names.Add("Navya");
names.Remove("Sowmya_0031");
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
Console.WriteLine($"My name is {names[0]}.");
Console.WriteLine($"I've added {names[2]} and {names[3]} to the list.");
```

Output:

```
Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!

Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!
Hello MAGGIE!
Hello NAVYA!
My name is <Sowmya_0031>.
I've added Felipe and Maggie to the list.
```

We're not allowed to access past the end of the list. we can check how long the list is using the Count property

```
var names = new List<string> { "<Sowmya_0031>", "Ana", "Felipe" };
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
Console.WriteLine();
names.Add("Maggie");
names.Add("Navya");
names.Remove("Sowmya_0031");
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
Console.WriteLine($"My name is {names[0]}.");
Console.WriteLine($"I've added {names[2]} and {names[3]} to the list.");
Console.WriteLine($"The list has {names.Count} people in it");
```

output:

```
Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!

Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!
Hello MAGGIE!
Hello NAVYA!
My name is <Sowmya_0031>.
I've added Felipe and Maggie to the list.
The list has 5 people in it
```

Search And Sort Lists:

Our samples use relatively small lists, but your applications may often create lists with many more elements, sometimes numbering in the thousands. To find elements in these larger collections, you need to search the list for different items. The IndexOf method searches for an item and returns the index of the item. If the item isn't in the list, IndexOf returns -1

```
var names = new List<string> { "<Sowmya_0031>", "Ana", "Felipe" };
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
var index = names.IndexOf("Felipe");
if (index != -1)
  Console.WriteLine($"The name {names[index]} is at index {index}");

var notFound = names.IndexOf("Not Found");
  Console.WriteLine($"When an item is not found, IndexOf returns {notFound}");
```

Output:

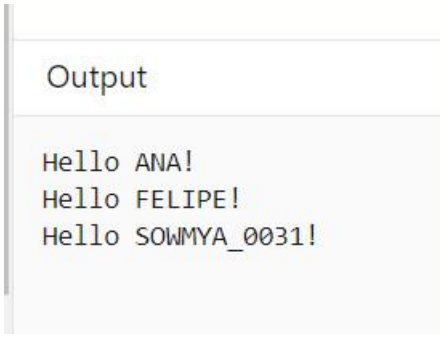
```
Output

Hello <SOWMYA_0031>!
Hello ANA!
Hello FELIPE!
The name Felipe is at index 2
When an item is not found, IndexOf returns -1
```

we may not know if an item is in the list, so we should always check the index returned by IndexOf. If it is -1, the item was not found.

The items in your list can be sorted as well. The Sort method sorts all the items in the list in their normal order (alphabetically for strings).

```
var names = new List<string> { "Sowmya_0031", "Ana", "Felipe" };
names.Sort();
foreach (var name in names)
{
  Console.WriteLine($"Hello {name.ToUpper()}!");
}
}
```

Output:

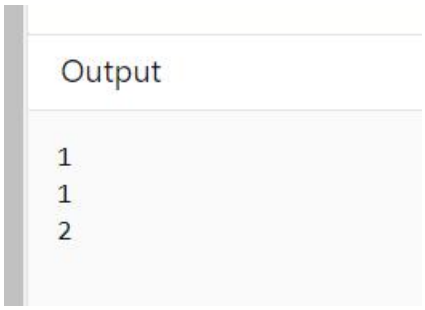


List Of Other Types:

We've been using the string type in lists so far. Let's make a List<T> using a different type. Let's build a set of numbers.

```
var fibonacciNumbers = new List<int> {1, 1};
var previous = fibonacciNumbers[fibonacciNumbers.Count - 1];
var previous2 = fibonacciNumbers[fibonacciNumbers.Count - 2];

fibonacciNumbers.Add(previous + previous2);

foreach(var item in fibonacciNumbers)
    Console.WriteLine(item);
```

Output:



That creates a list of integers, and sets the first two integers to the value 1. The Fibonacci Sequence, a sequence of numbers, starts with two 1s. Each next Fibonacci number is found by taking the sum of the previous two numbers.

Code to generate the first 20 numbers in the sequence:

```
var fibonacciNumbers = new List<int> {1, 1};

while (fibonacciNumbers.Count < 20)
{
    var previous = fibonacciNumbers[fibonacciNumbers.Count - 1];
    var previous2 = fibonacciNumbers[fibonacciNumbers.Count - 2];

    fibonacciNumbers.Add(previous + previous2);
```

```
}
foreach(var item in fibonacciNumbers)
    Console.WriteLine(item);
```

Output:

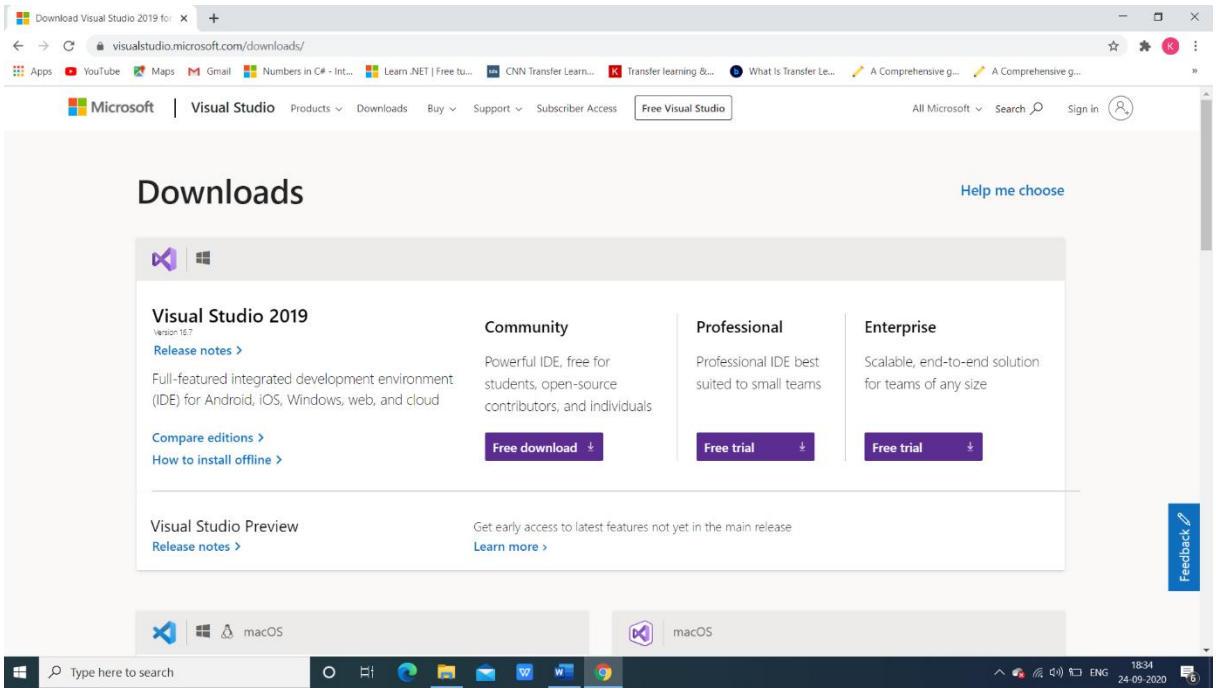
```
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
```

List collections completed.

Week 4 (16-09-2020)

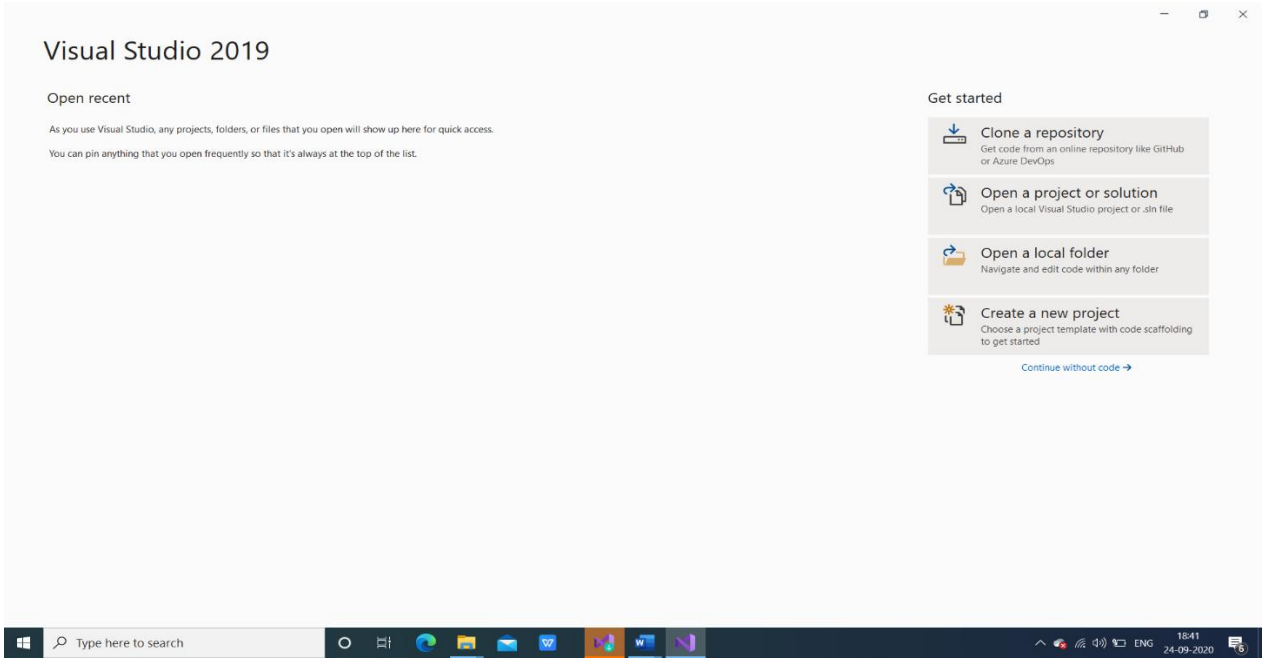
Install Visual Studio 2019 from https://visualstudio.microsoft.com/downloads/

Choose Community – Free Download



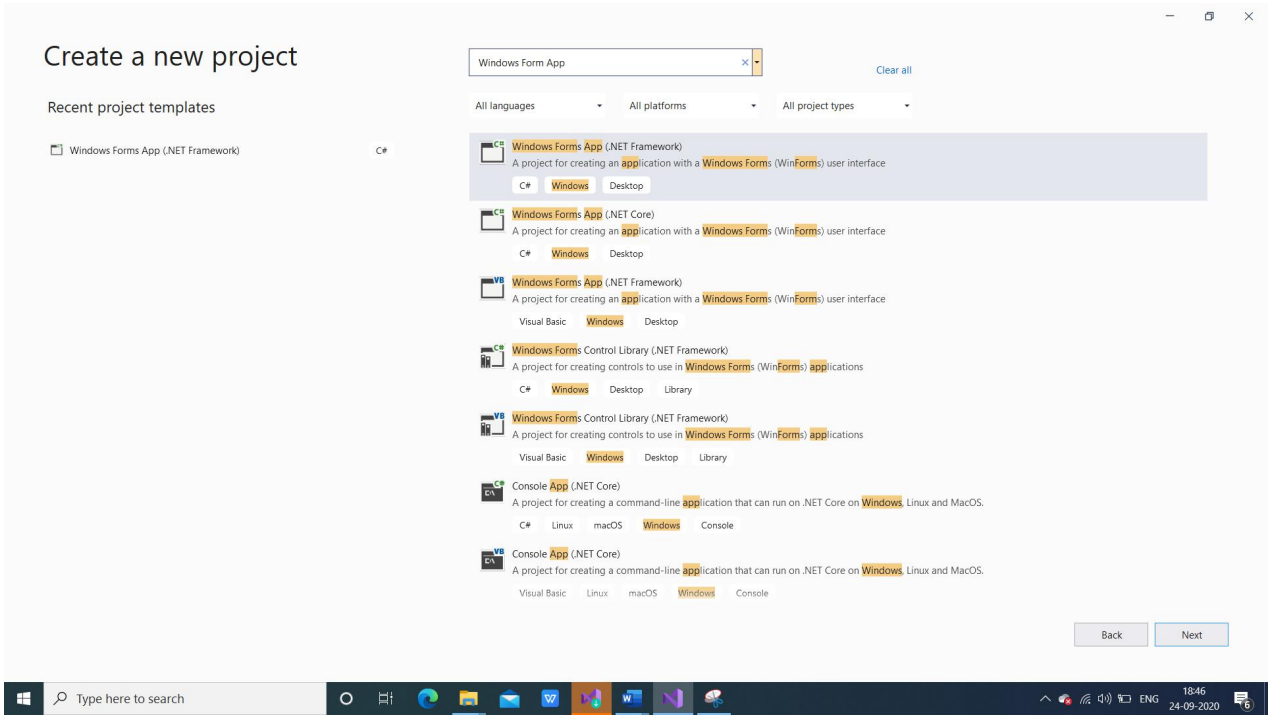
1)Open Visual Studio 2019

2)Select “Create a new poject”

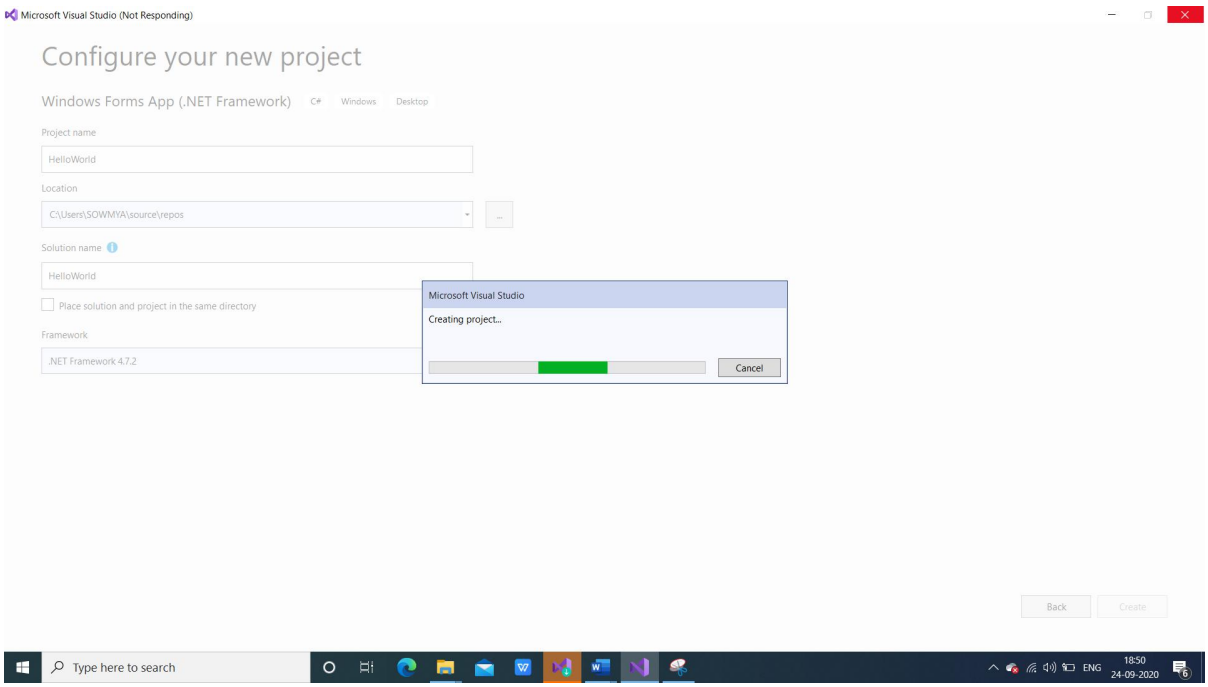


3) It is displayed

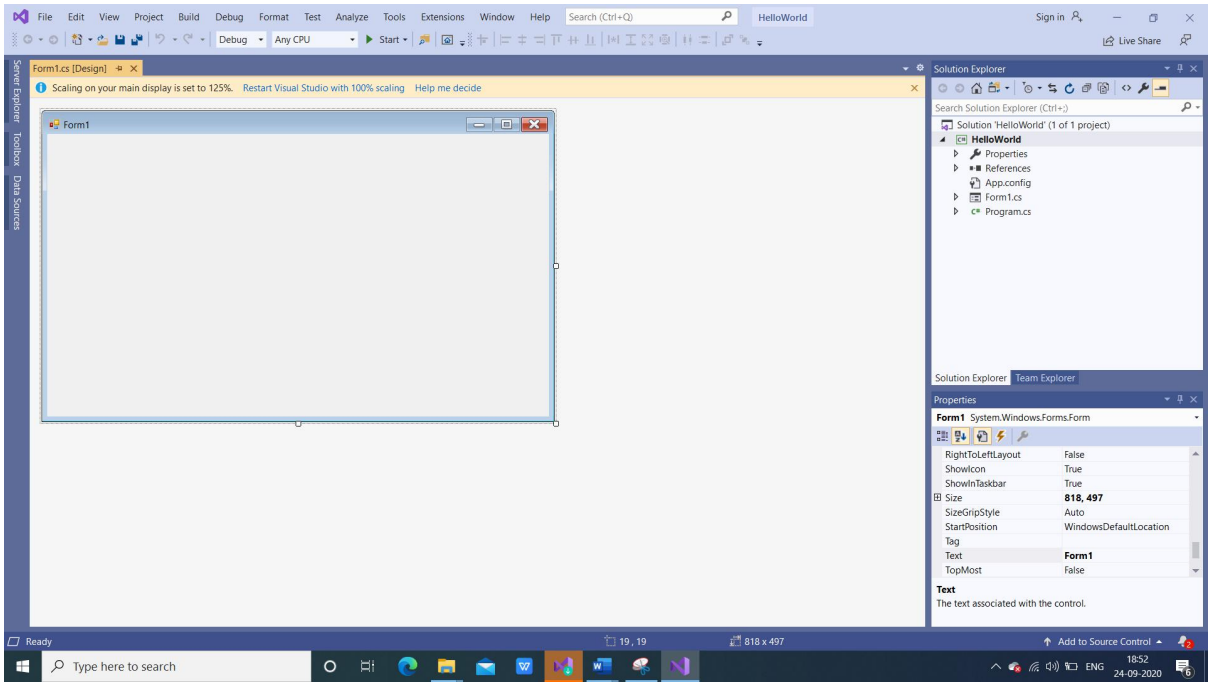
4) In the search bar i.e.,"Search for templates",Type "Windows Form App",Choose Windows Form App(.NET Framework) and Click on Next.



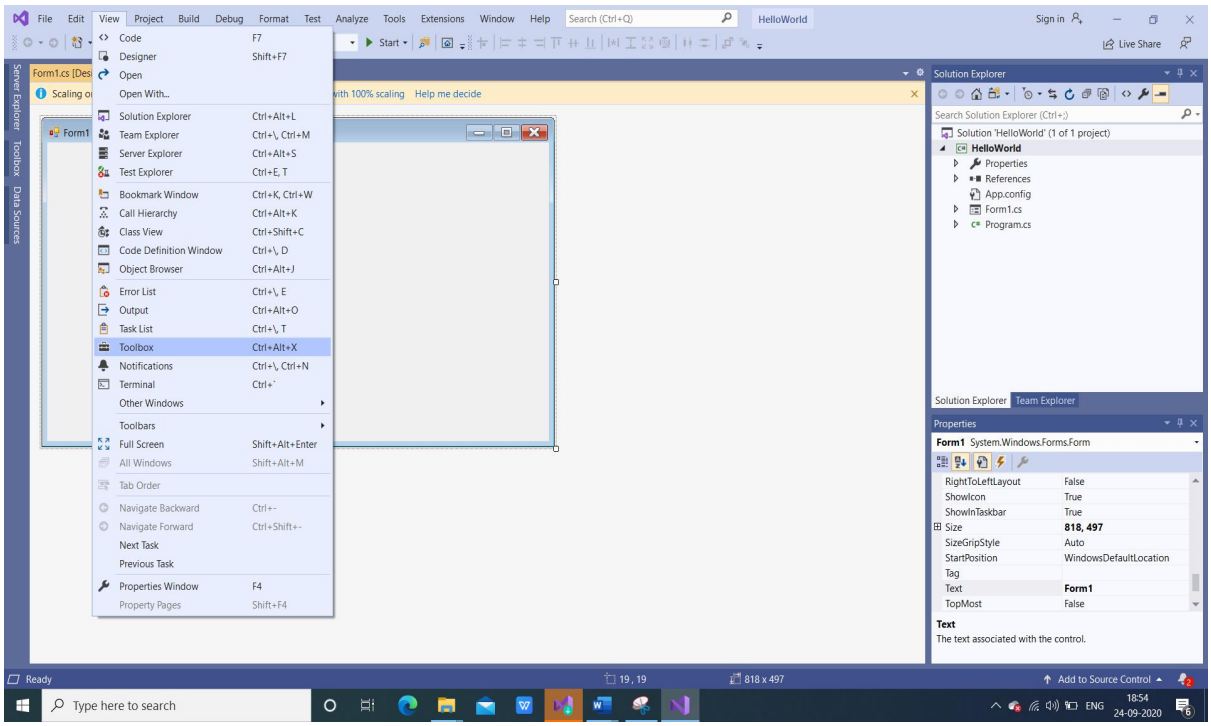
5)Under Configure your new Project, Project Name is changed to "HelloWorld" and click on Create.



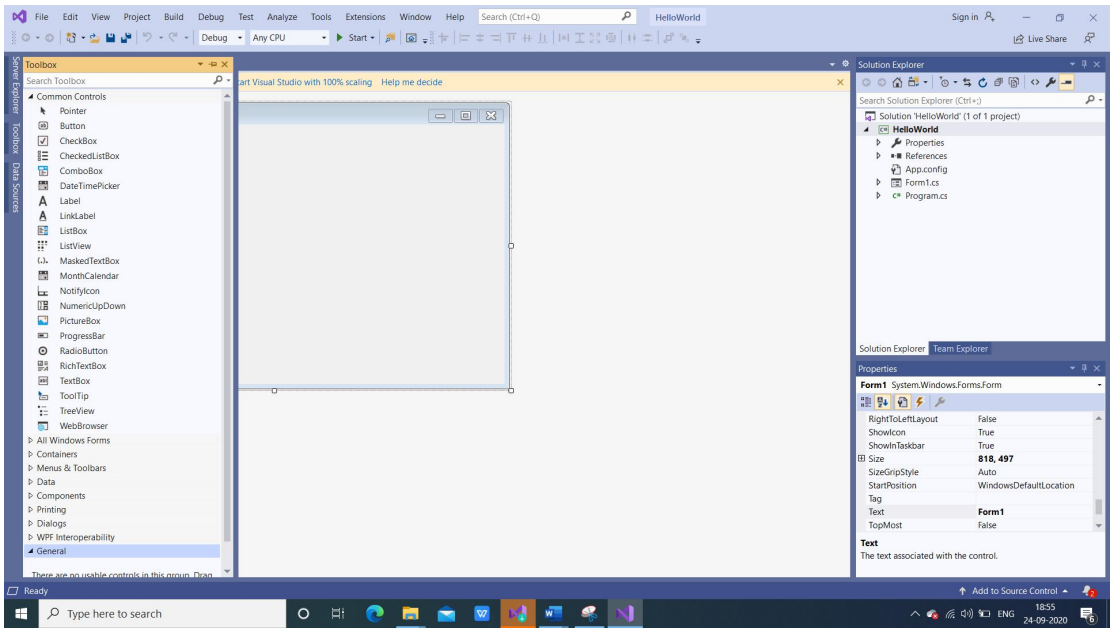
6) Below is the New Windows Form App:



7) To View the Toolbox,Click on View and choose Tool box:



8) Click on Common controls:



9) COMMON CONTROLS :

Pointer : This tool is selected by default when any Toolbox tab opens. It cannot be deleted. The pointer enables you to drag objects on to the design view surface, resize them, and re-position them on the page or form.

Button: Button Control in Windows form than it allows the user to click it to perform an action. It can display both text and images. when the button is clicked, it looks as if it is being pushed in and released.

CheckBox : It indicates whether a particular condition is On or Off. This window control is commonly used to present a Yes/No as well as True/False selection to the user. While using this window control, You can also use checkbox controls in groups to display multiple choices from which the user can select one or more.

CheckedListBox : Display a scrollable list of items, each accompanied by a checkbox.

ComboBox : This control is basically used for displaying data in a drop-down manner. Generally, This control appears in two parts: the top part is a text box that allows the user to type a list item and the second part is a list box that displays a list of items in which the user can select one.

DateTimePicker: Display a graphical calendar to allow a user to select a date or a time.

Label : Windows Forms Label controls basically used to display text or images that cannot be edit by the user. They are used to identify the object on a form – to provide a description of what to certain control will do if clicked, for example, or to display information in response to the time event or process in your application

LinkLable : Display Text as a Web-Style link and triggers an event when the user clicks the special text. Usually, the text is a link to another window or a WebSite.

ListBox : A ListBox is a useful tool that helps you to display a list of several items from which you can select one or more items. A scrollbar is automatically added to the list box when the number of items in the list box increases.

ListView : Displays items in one of four different views. Views include text-only, Text with small icons, Text with large icons and a report view.

MaskedTextBox : Uses a mask to distinguish between proper and improper user inputs

MonthCalendar: Display a graphical calendar to allow a user to select a range of dates.

NotifyIcon: Display an icon in the status notification areas of the taskbar that represents an application running in the background.

NumericUpDown: Display a list of numerical that user can scroll through with Up and Down buttons.

PictureBox: With the help of **System.Windows.Forms.PictureBox** control, It is possible to load as well as display a picture on a form at design time by setting the System.Windows.Forms.PictureBox.image property to a valid picture.

ProgressBar : ProgressBar basically used for showing the progress of some operations by displaying rectangles in the horizontal bar. ProgressBar also has some main properties which are value, minimum and maximum. You can also use minimum and maximum properties to set the minimum and maximum values the progress bar can display. To change the display, you can write code as well as set the value property. As a result, If the maximum property is set to 100, the minimum property is set to 10.

RadioButton : Radio buttons are also checkboxes , Basically, RadioButtons are round as against the checkboxes which are square.Therefore, Radio buttons are used mostly in groups, While checkboxes are used individually.

Rich TextBox : Enable text to be displayed with formatting in plain text or Rich Text Format(RTF). For displaying multiple types of formatted text, we should use the RichTextBox control.

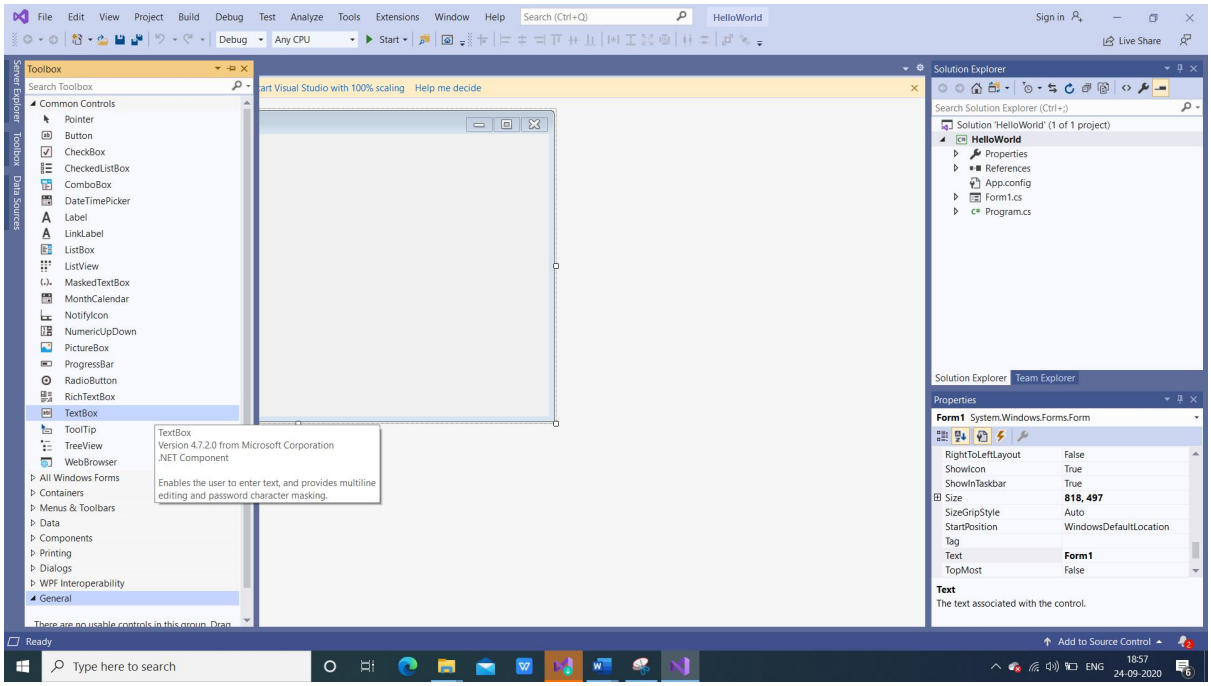
TextBox : TextBox is used to get input from the user or to display text. Basically, It is used for editable text, although it can also be made read-only. Text Boxes can display multiple lines as well as it wraps text to the size of the control, and add basic formatting.

ToolTip : Display information when the user moves the pointer over an associated control.

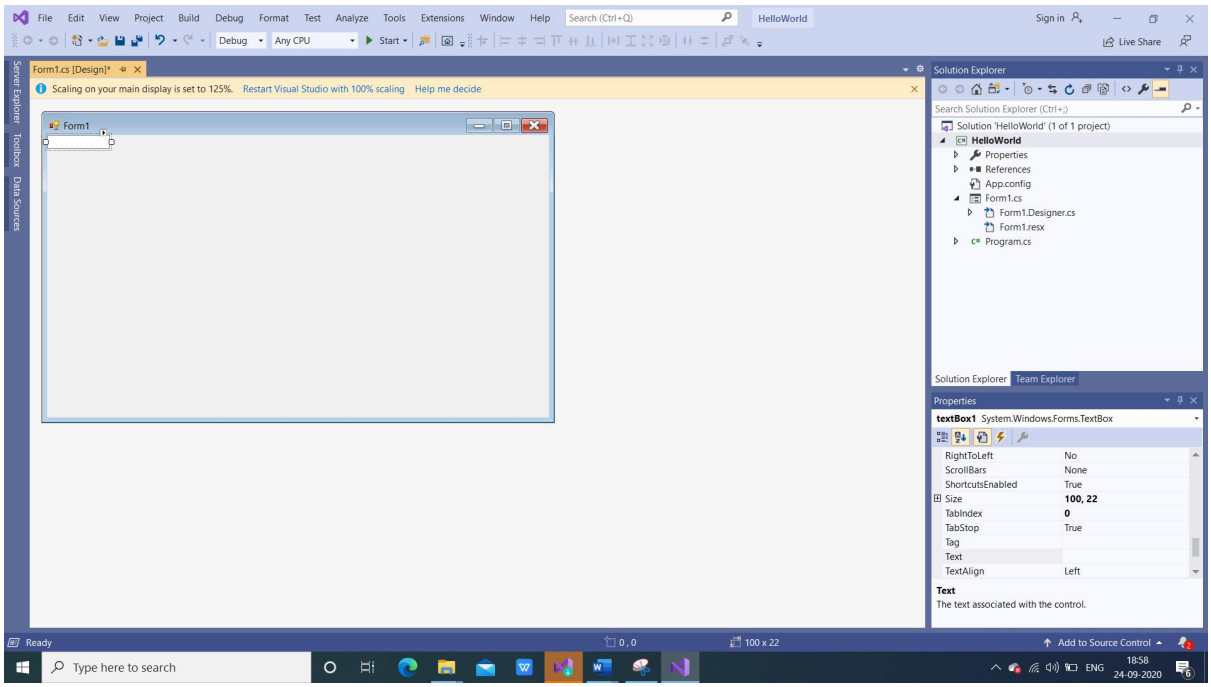
TreeView : Displays a hierarchical collection of node objects which can consist of text with optional check boxes and icons.

WebBrowser: Enables the user to browse webpages inside your form.

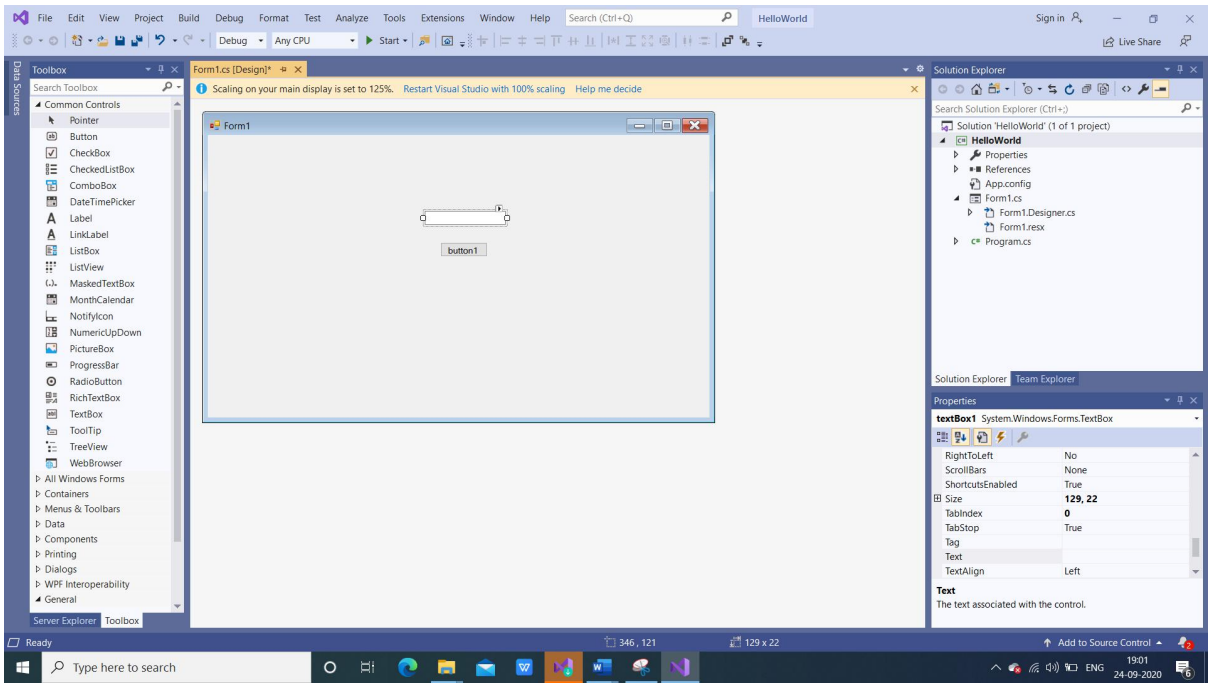
10) Choose “Textbox” from the Common controls - Double click on Text box control.



11)We can see a Text box under Form 1:

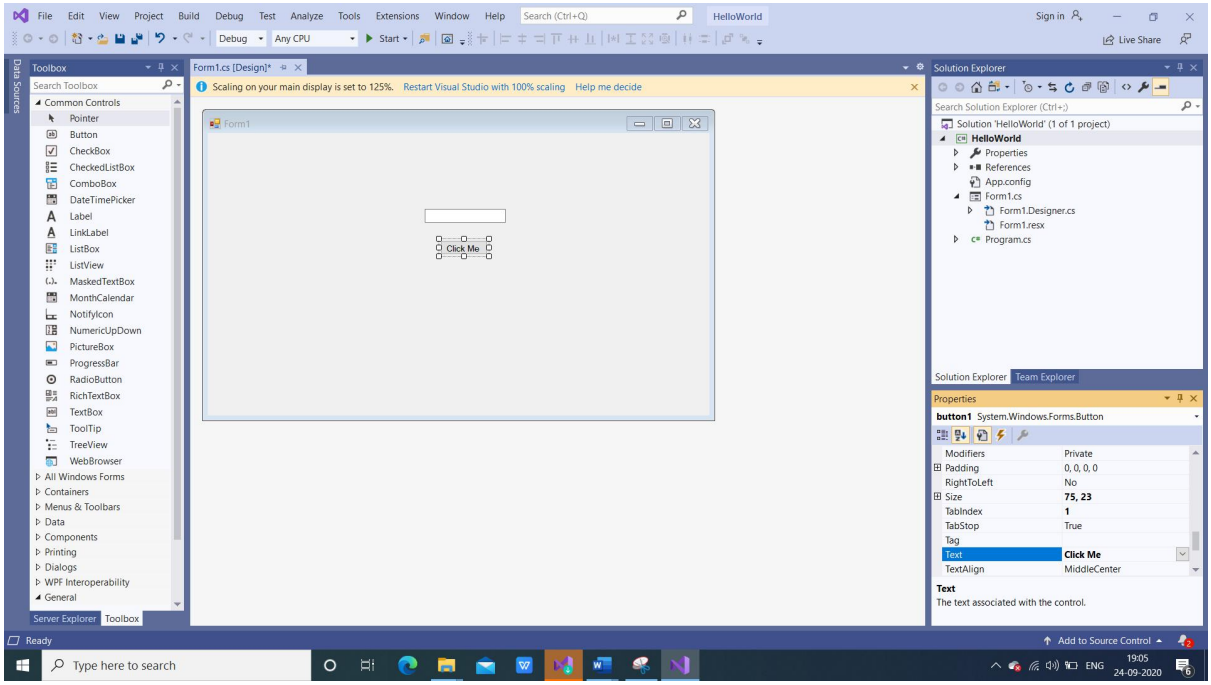


12) Choose “Button” from the Common controls - Double click on Button control

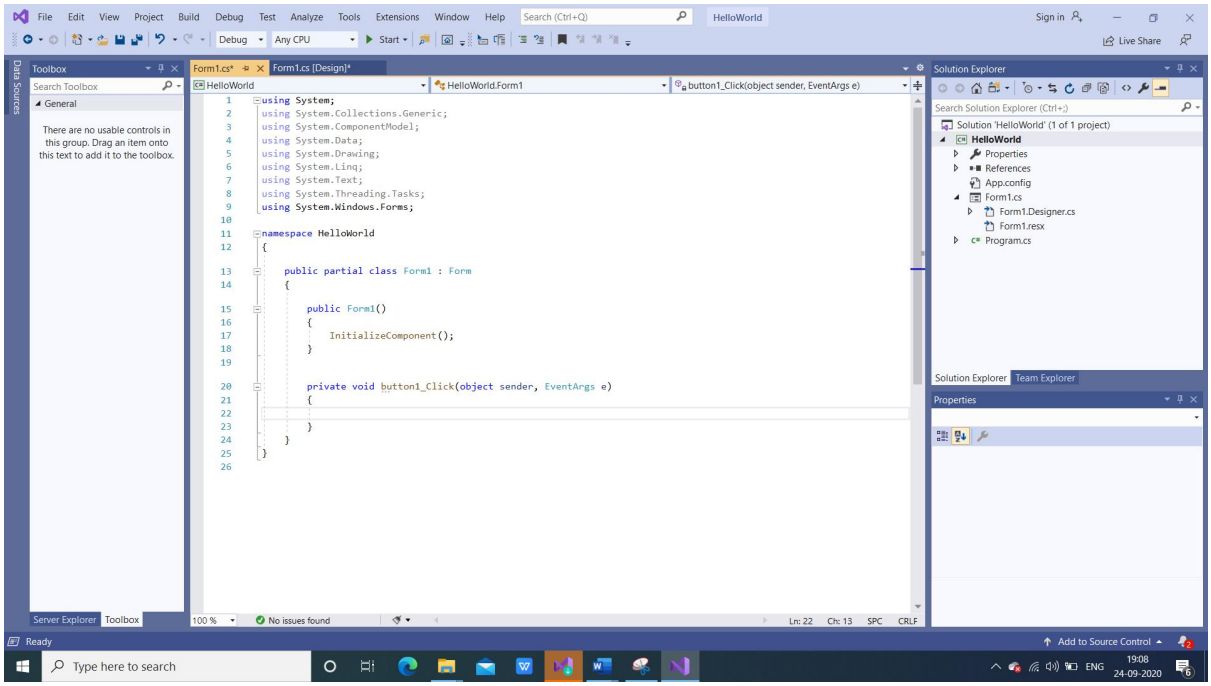


13) Change the button1 name to “Click Me”. You can change the button1 name under “Properties” – Right side corner of application.

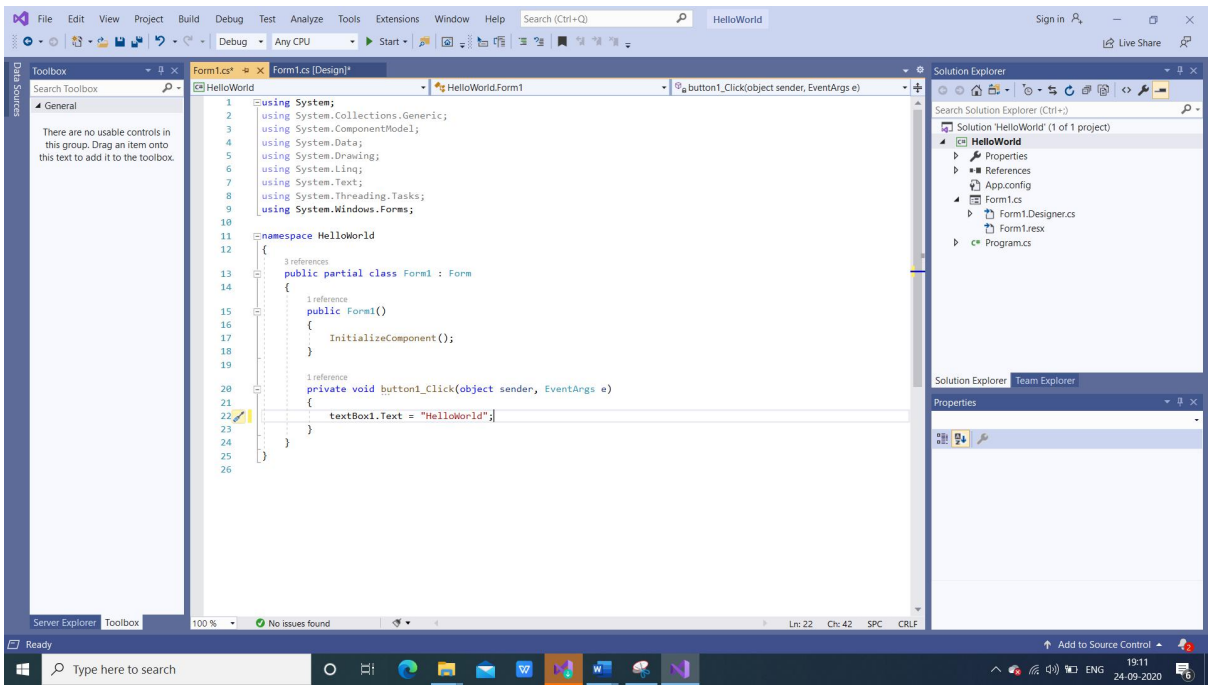
Step 1 : Click on the button1
Step 2: Beside “Text” you can find name as button1,change it to Click Me.



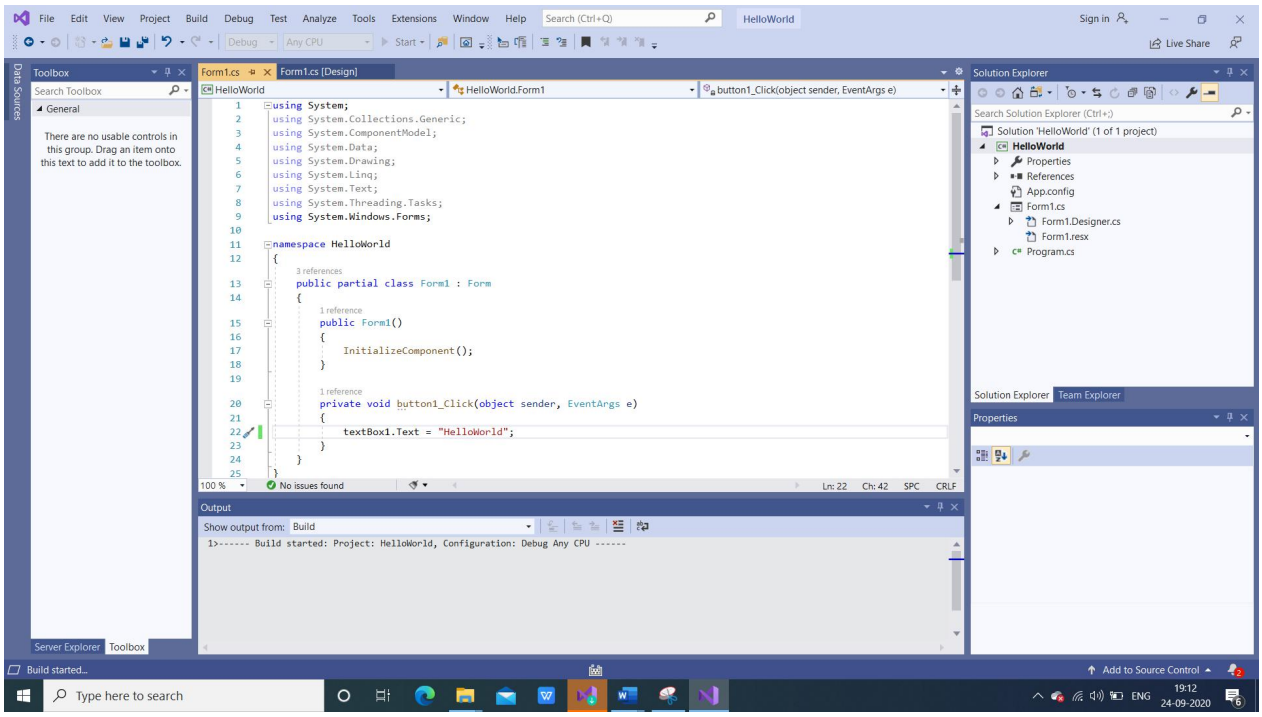
14) Task 1: If user clicks on Button i.e., Click Me, Hello World must be displayed inside the textbox : Now, Double on button i.e., Click Me.

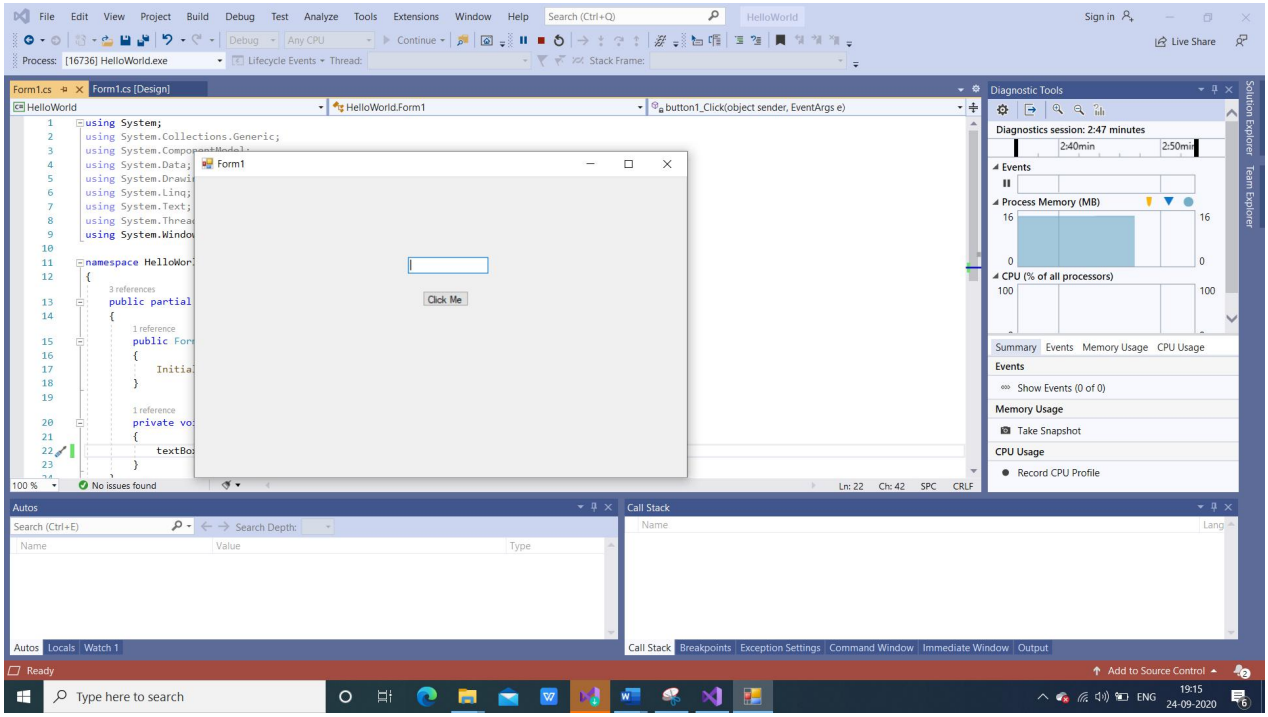


15) Input textbox1.text = “Hello world”

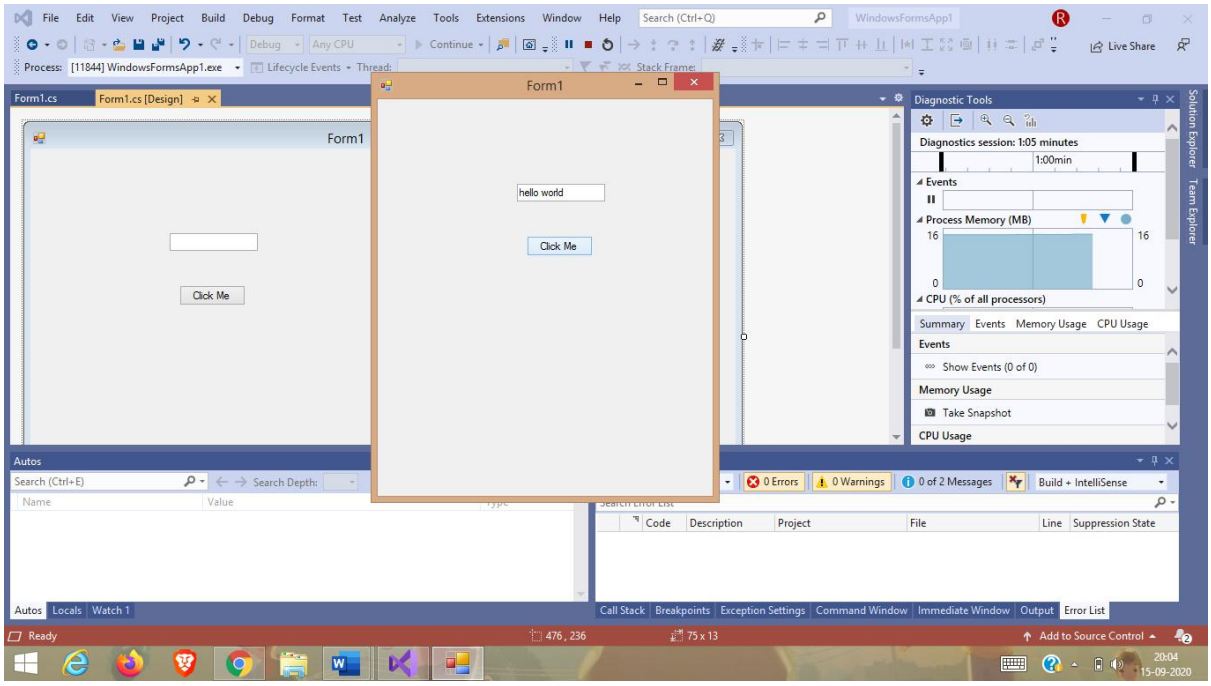


16)Click on Save and Click on Start Run.

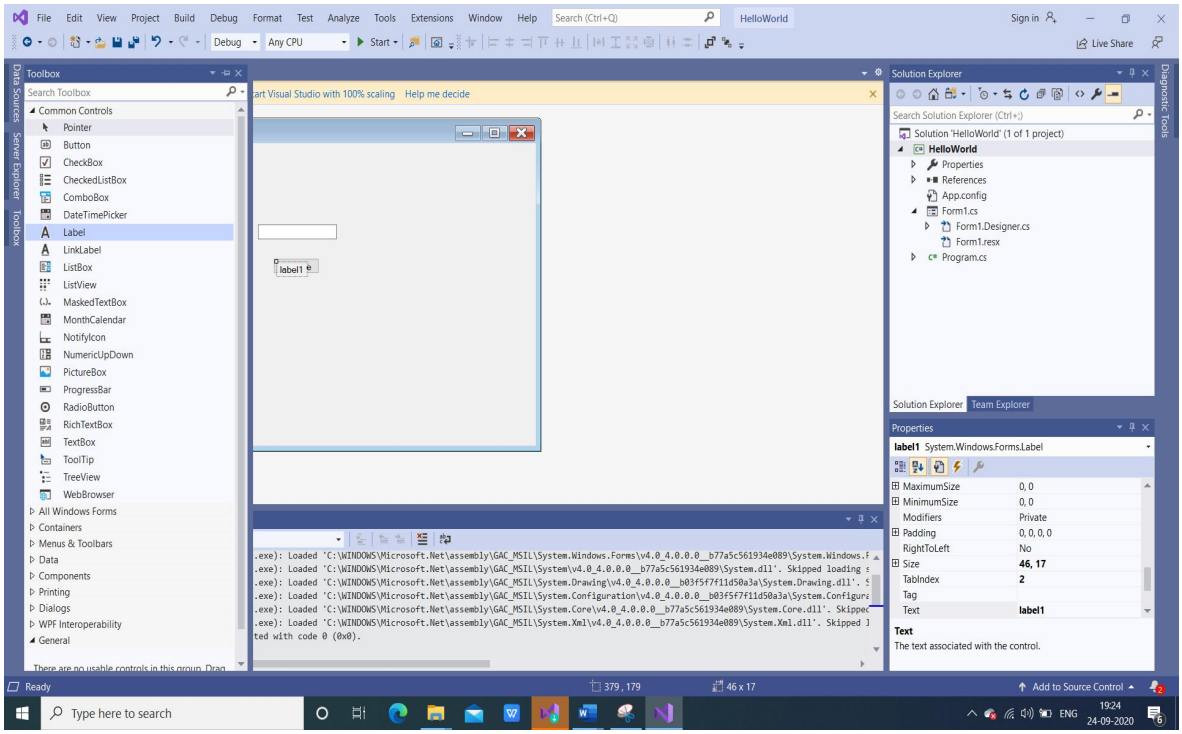




17)A pop up Windows form application will be displayed. Click on buttoni.e.Click ME, You can see the text “Hello World “ inside the Textbox i.e., The Output – hello world



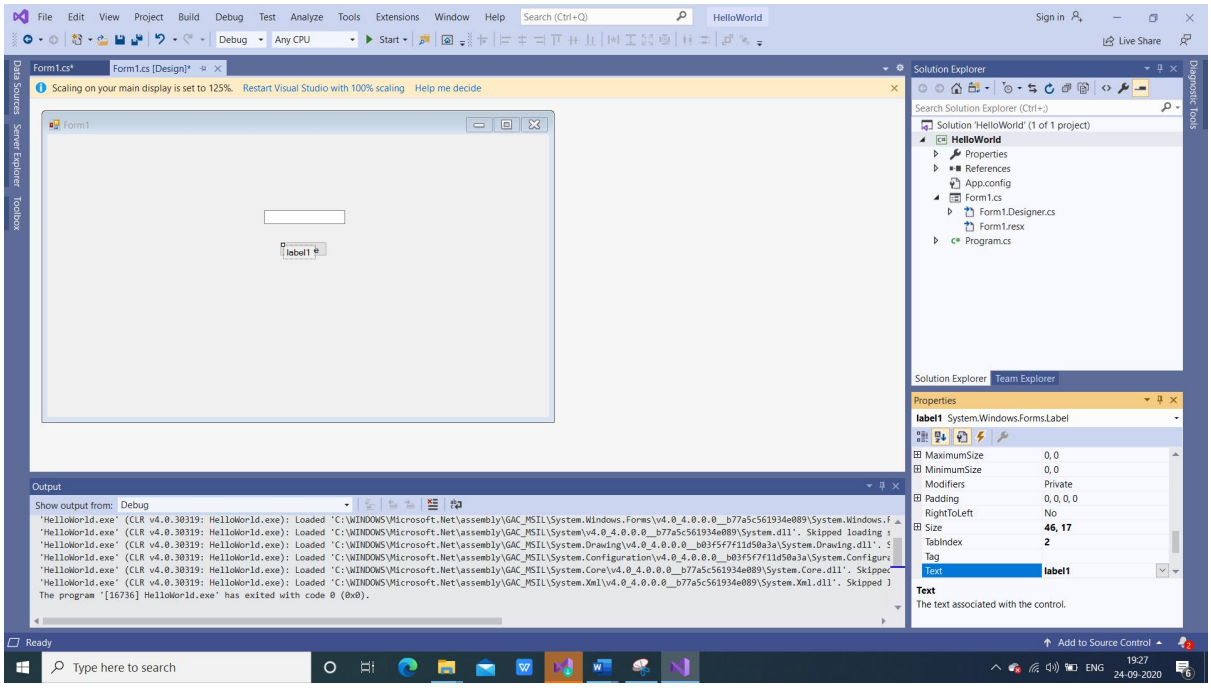
18) Choose Label from Common Controls i.e., Double click on Label

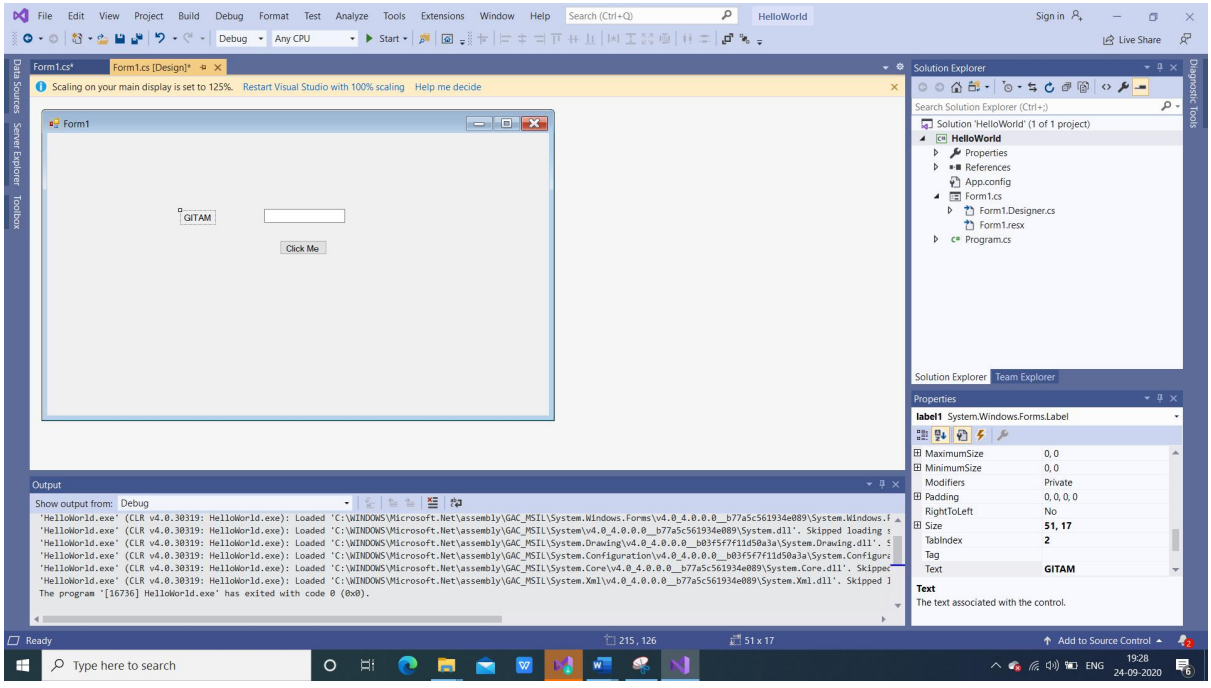


19) Change the label1 name to “GITAM”. You can change the label1 name under “Properties” – Right side corner of application.

Step 1: Click on the label1

Step 2: Beside “Text” you can find name as button1, change it to GITAM.

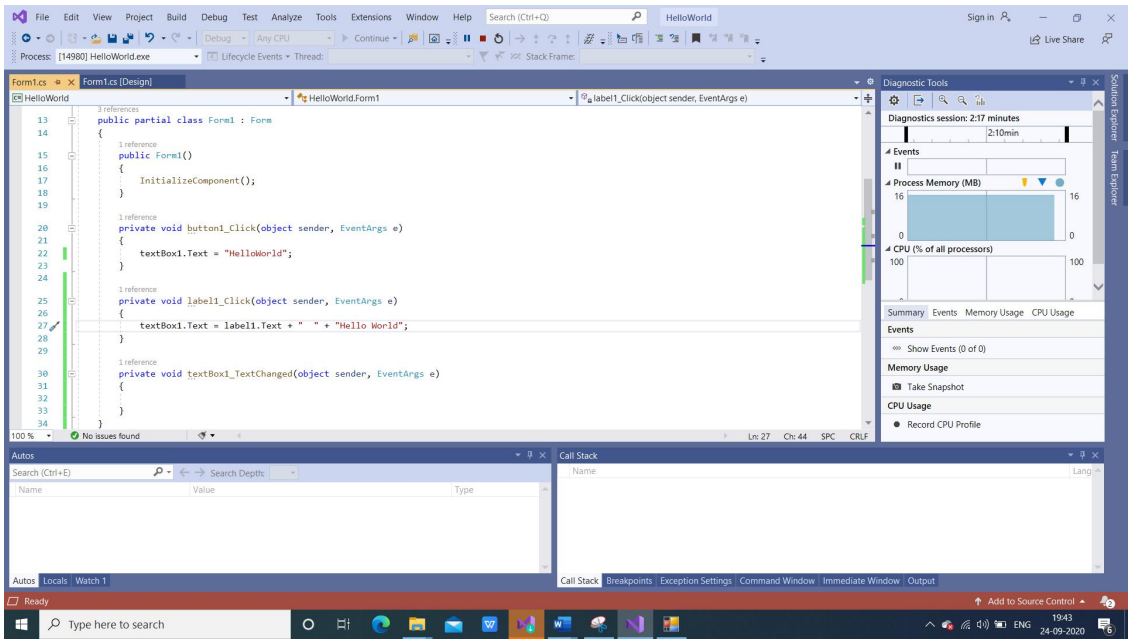




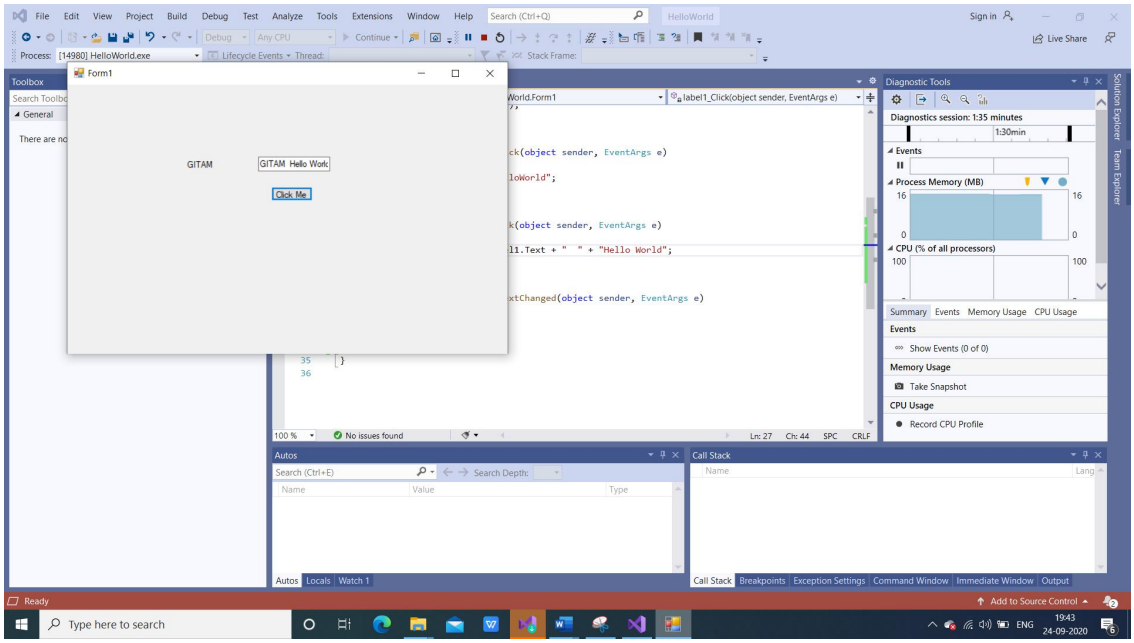
20)Task 2: If user clicks on Button i.e., Click Me, GITAM Hello World must be displayed inside the textbox://textBox1.Text = "hello world";

textbox1.text = label1.text + “ “ Hello World

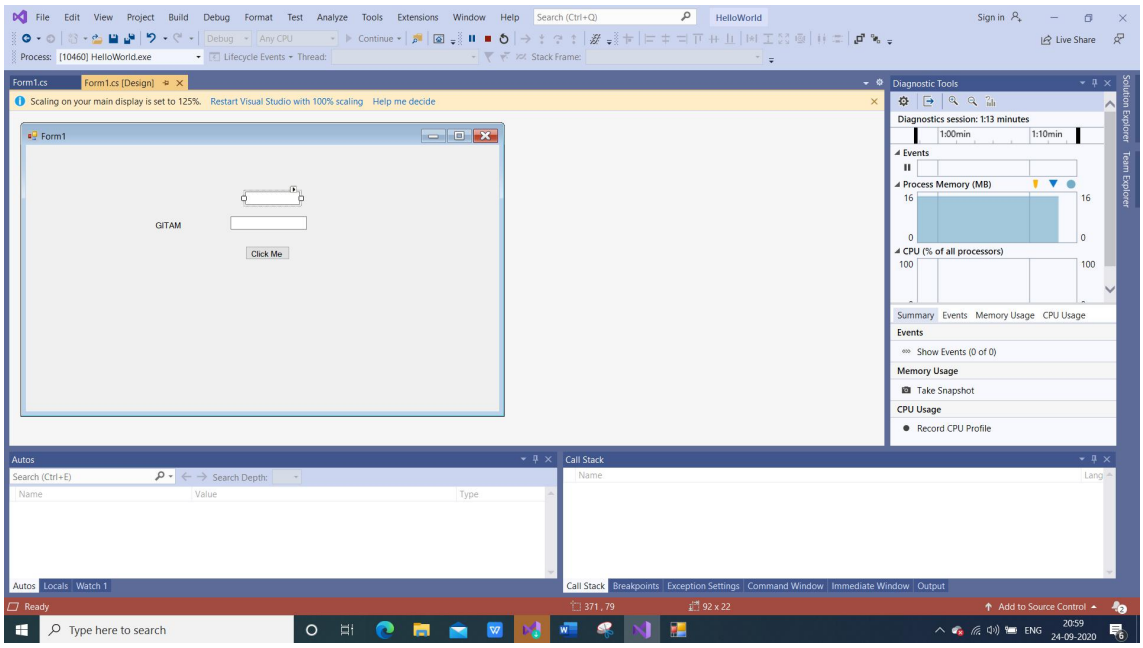
Click on Save and Click on Start Run.



21)A pop up Windows form application will be displayed. Click on button i.e.,Click Me, You can see the text “GITAM Hello World “ inside the Textbox i.e., The Output – GITAM hello world as below:

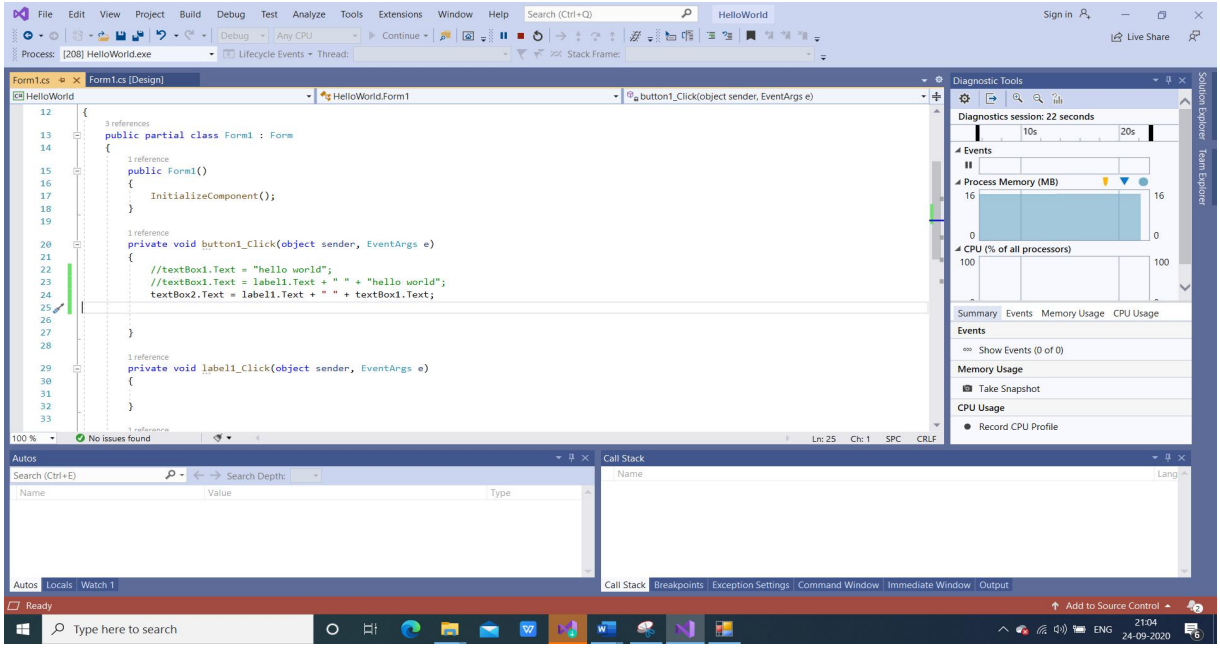


22)Task 3: Give any data in the text box1, If User clicks on “Click me” button,Output must be GITAM and textbox1 data. For example,If user enters Visakhapatnam in text box1,The output should be GITAM Visakhapatnam.
Choose Another “Textbox” from the Common controls - Double click on Text box control.



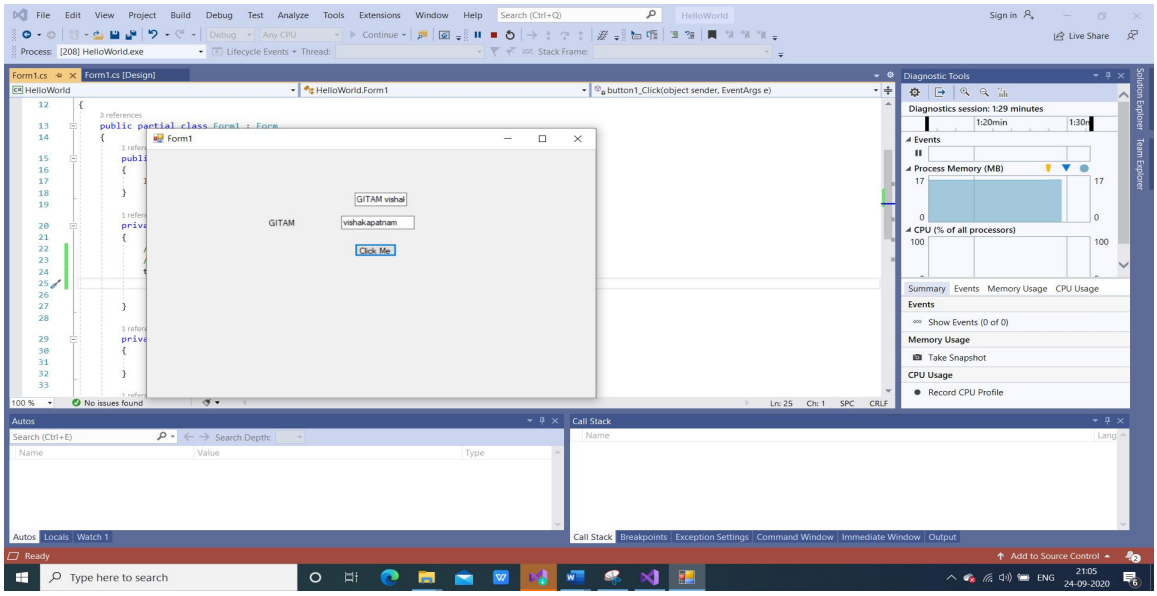
23)Double click on “Click Me button”

```
//textBox1.Text = "hello world";
//textBox1.Text = label1.Text + " " + "hello world";
textBox2.Text = label1.Text + " " + textBox1.Text;
```



24)Click on Save and Start Run

25)A pop up Windows form application will be displayed. : Enter Visakhapatnam in the text box1.,If user enters Visakhapatnam in text box1,The output should be GITAM Visakhapatnam as below:

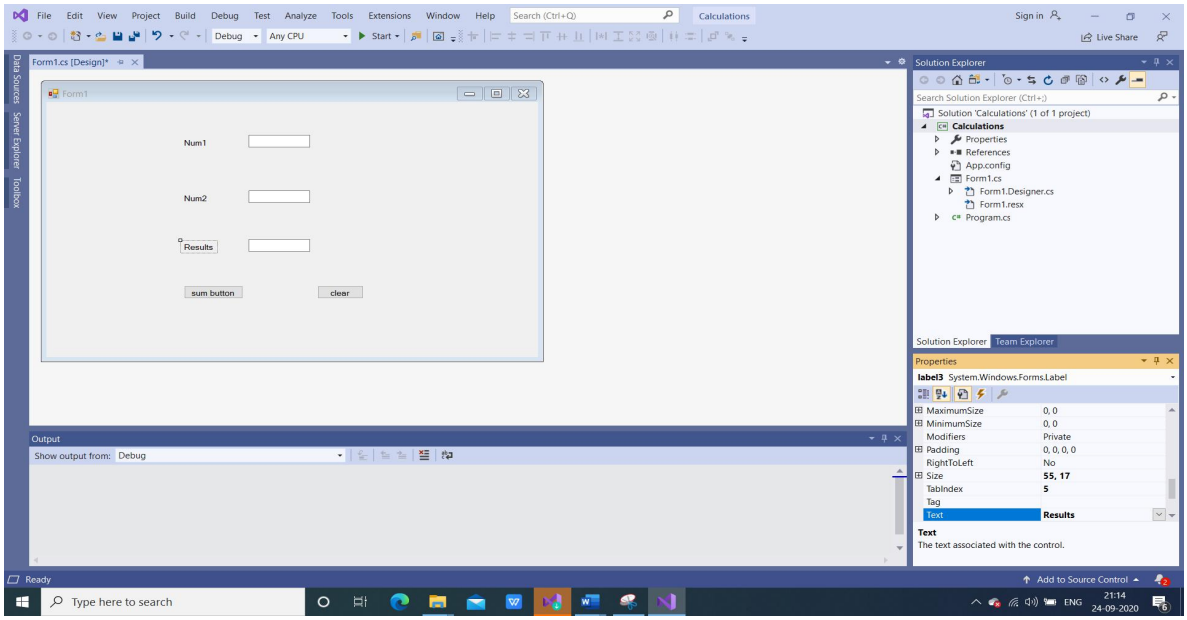


26) Task 4 : Calculator Application:

Create Three Text Boxes.

Create Three label Buttons,Re-name label1 as Num1, Re-name label2 as Num2, Re-name label3 as RESULT.

Create Two button's .Rename button1 as Sum Button and button2 as Clear.

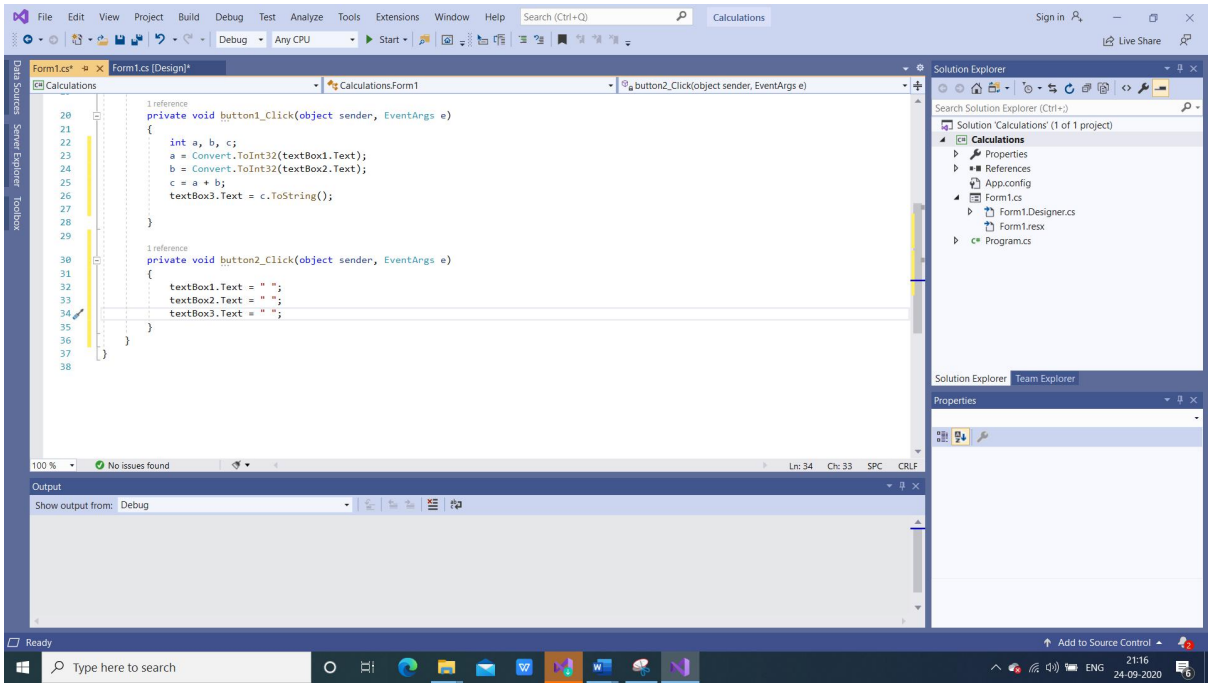


27)As logic will be in Sum Button,Double click on Sum Button.

```
int a, b, c;
a = Convert.ToInt32(textBox1.Text);
b = Convert.ToInt32(textBox2.Text);
c = a + b;
textBox3.Text = c.ToString();
```

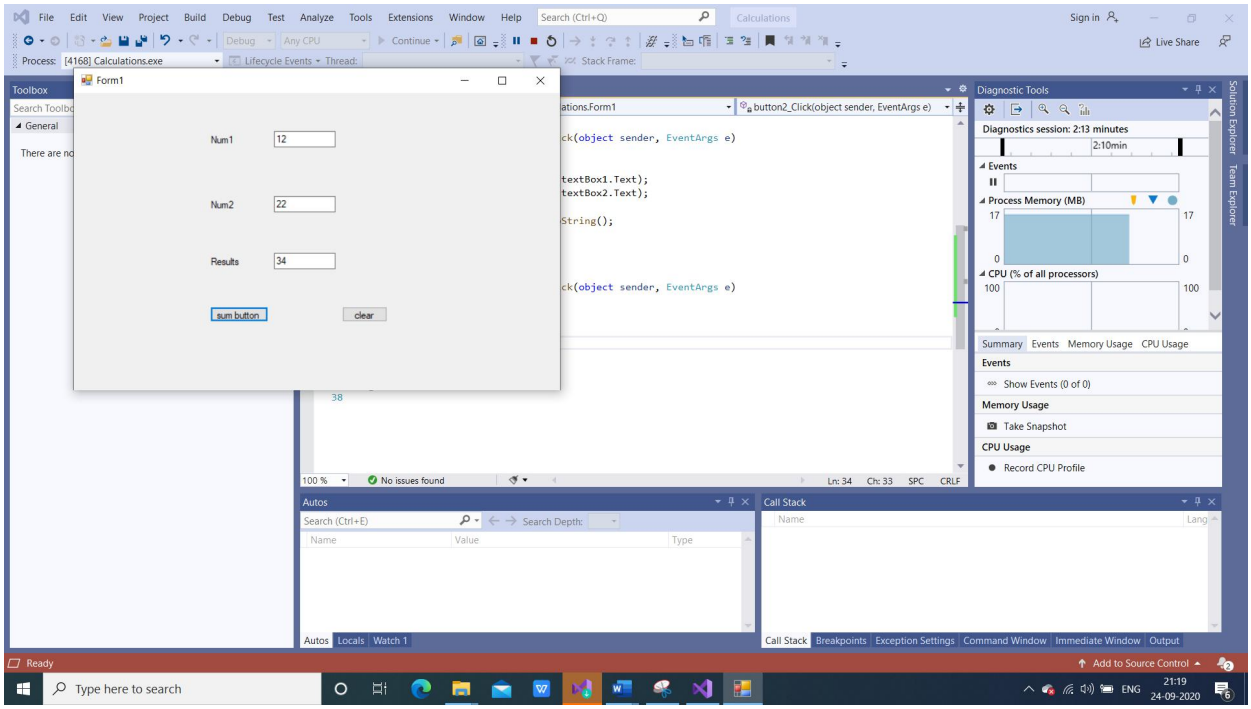
Double click on Clear button.

```
textBox1.Text = " ";
textBox2.Text = " ";
textBox3.Text = " ";
```

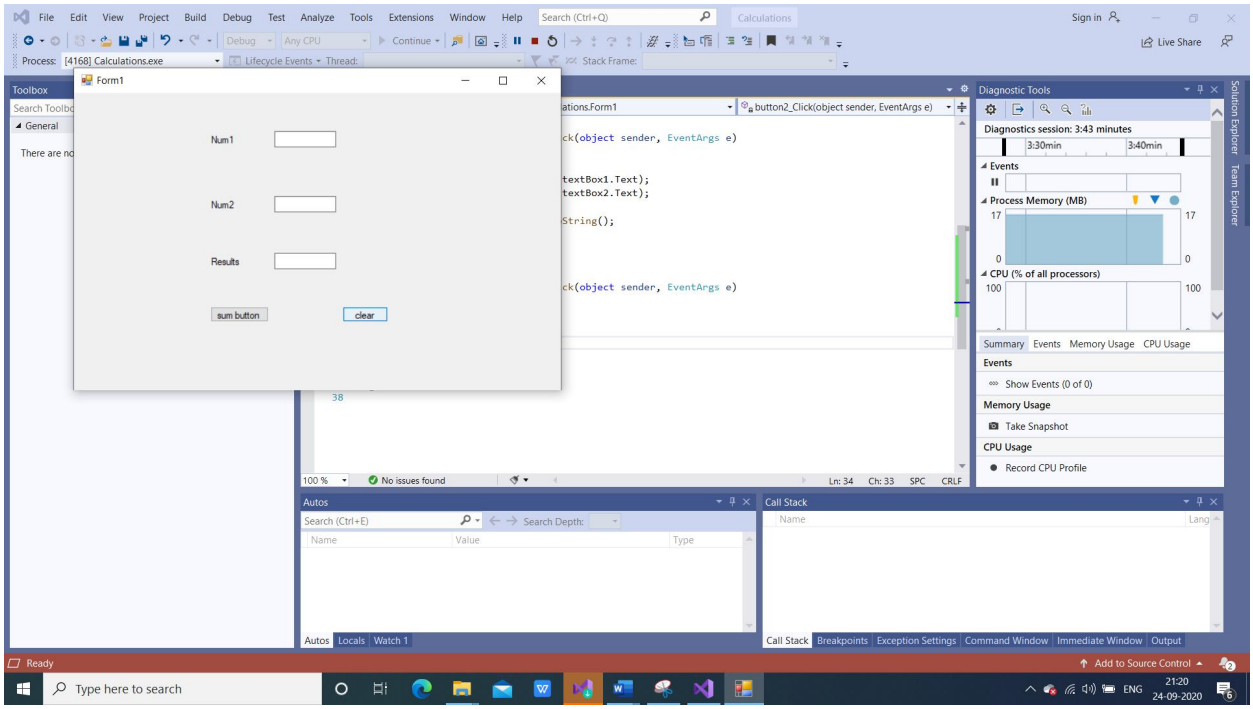


28) Click on Save and Start Run

29)A pop up Windows form application will be displayed. : Enter Num1 and Num2 abd Click on Sum Button,You can find the output in Result:



Click on Clear,you can observe all fields will be cleared :

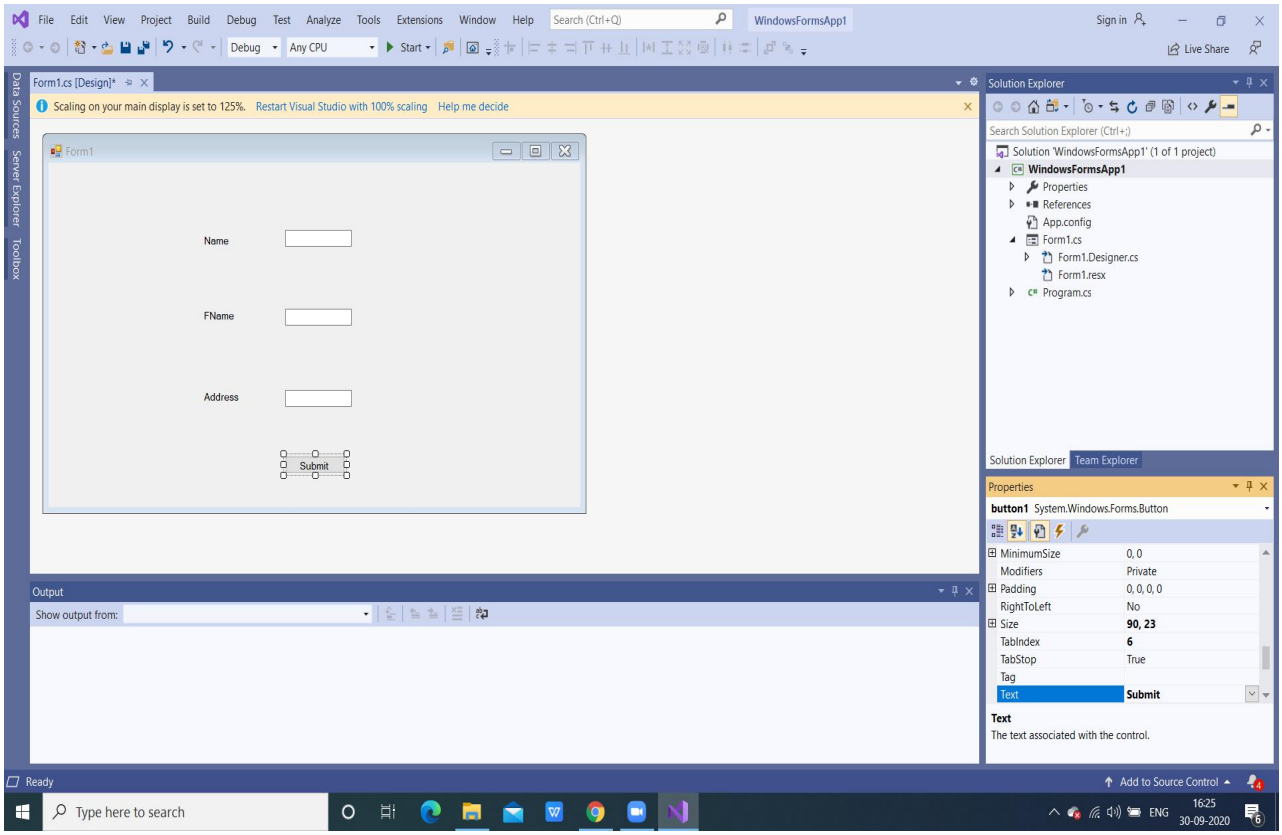


WEEK 5 (30-09-2020)

Pass data from one form to another form in Windows Forms applications using C#.

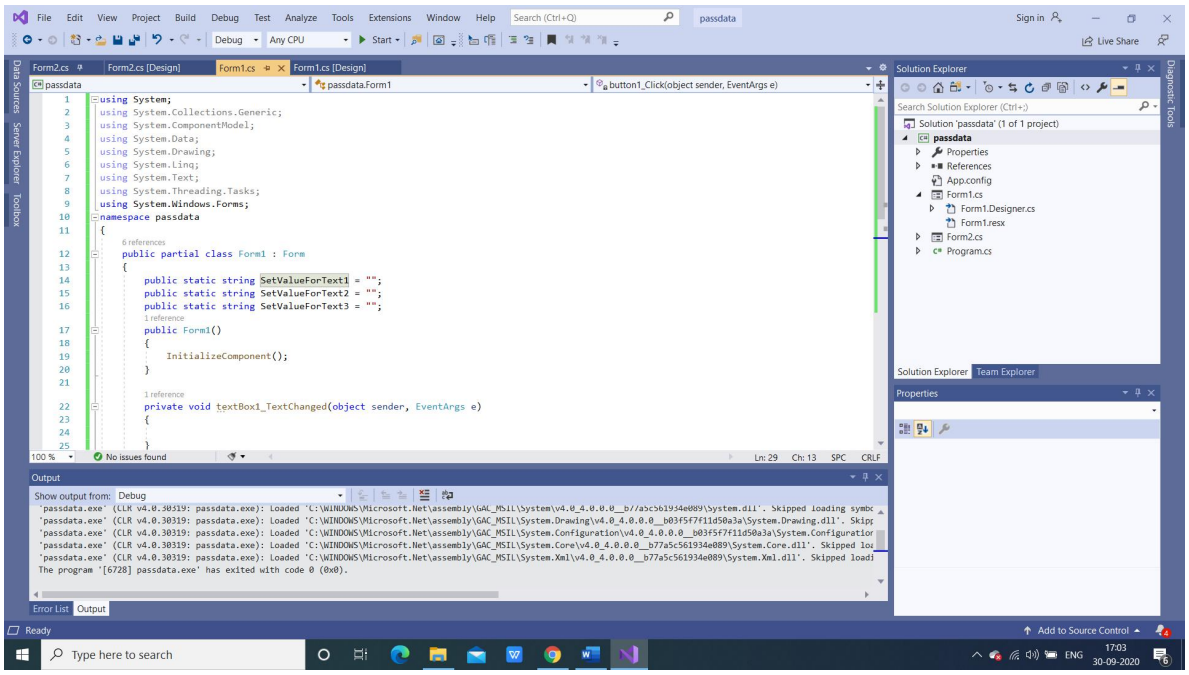
1) Create a new project and then select C# Windows Forms Application then click Ok

2) Drag and drop a Label and a TextBox from the Toolbox. We created a Form with 3 Labels and 3 TextBoxes.
Change the label names and textbox names. we can change the name under "Properties" – Right side corner of application.

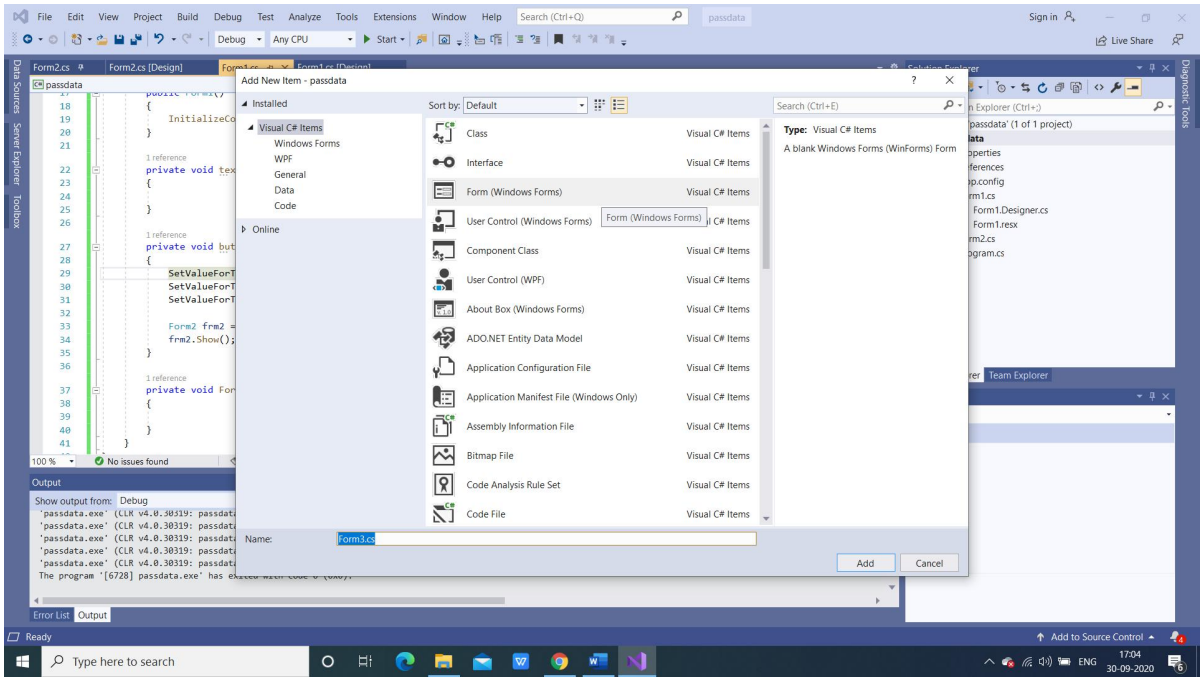


3) We have a Name, Fname and Address Label on the form. So we use three global variables Write the following code in the Form1.cs :

```
public static string SetValueForText1 = "";
public static string SetValueForText2 = "";
public static string SetValueForText3 = "";
```

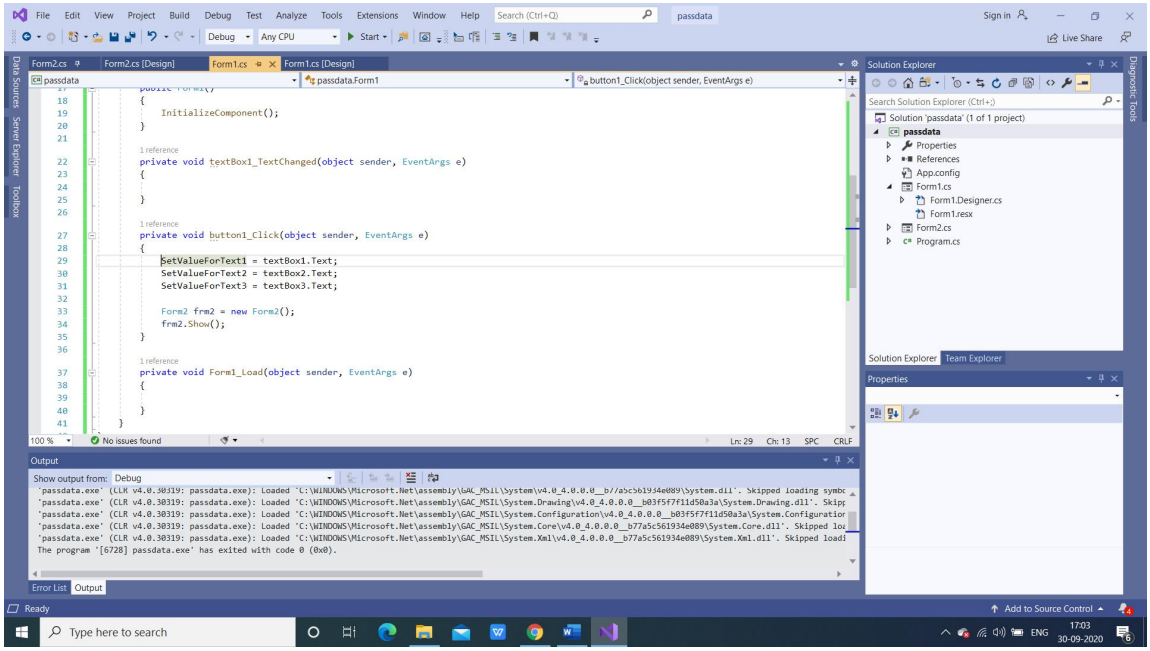


4) Add another Windows Forms form using Project --> Add Windows Form then click on Add.

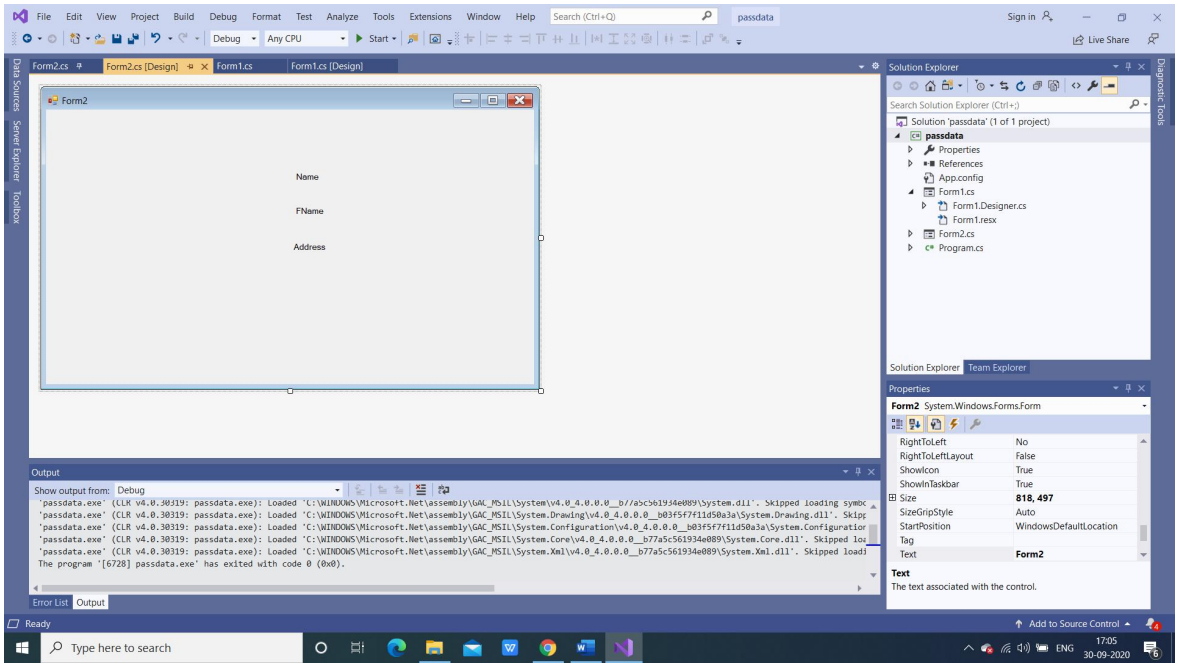


5) After creating the form double-click on the Submit button on the Windows Form1 and write the code:

```
SetValueForText1 = textBox1.Text;
SetValueForText2 = textBox2.Text;
SetValueForText3 = textBox3.Text;

Form2 frm2 = new Form2();
frm2.show();
```

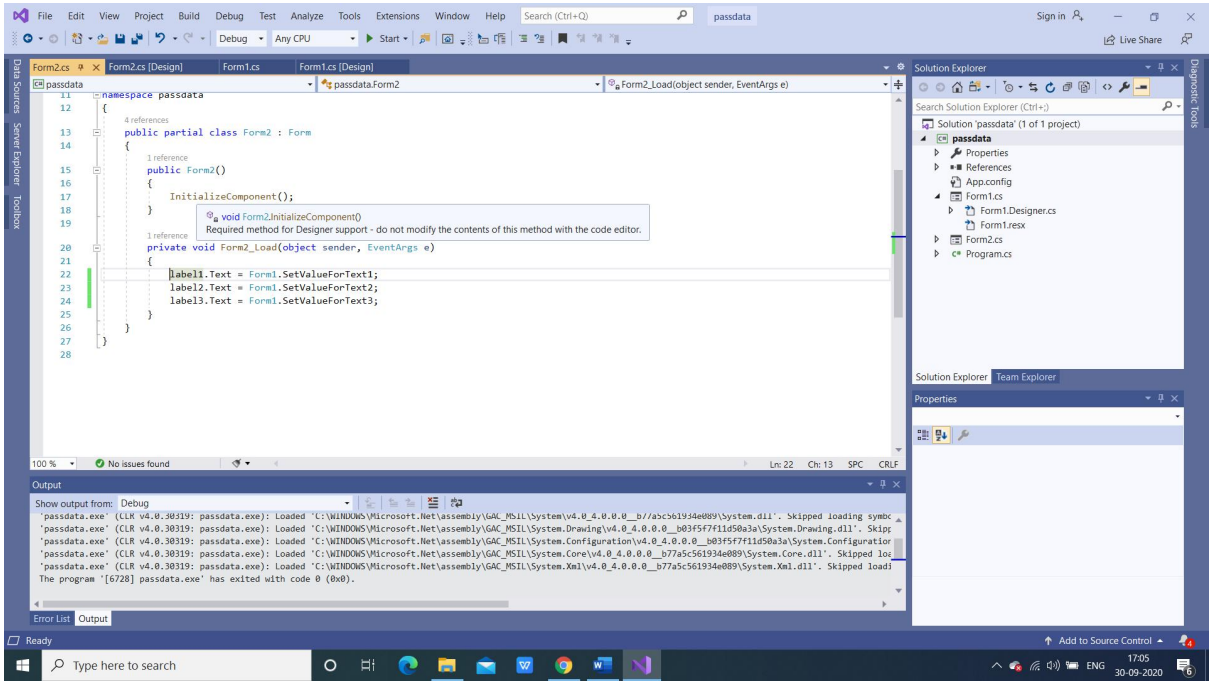


6) Drag and Drop 3 Labels from the Toolbox onto Form2 and rename them.



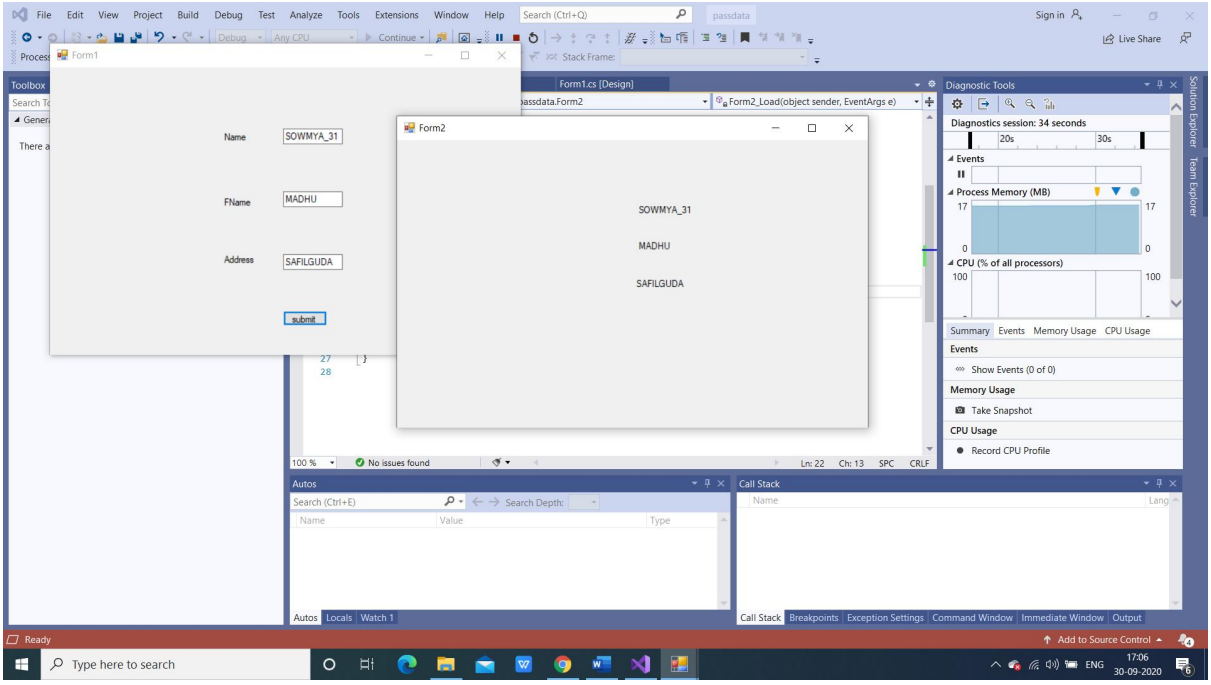
7) Double-click on Form2 and write the code:

```
label1.Text = Form1.SetValueForText1;
label2.Text = Form1.SetValueForText2;
label3.Text = Form1.SetValueForText3;
```



8) Now save and run the application.

Fill in Form1 and click on Submit. The data will pass from Form1 to Form2.

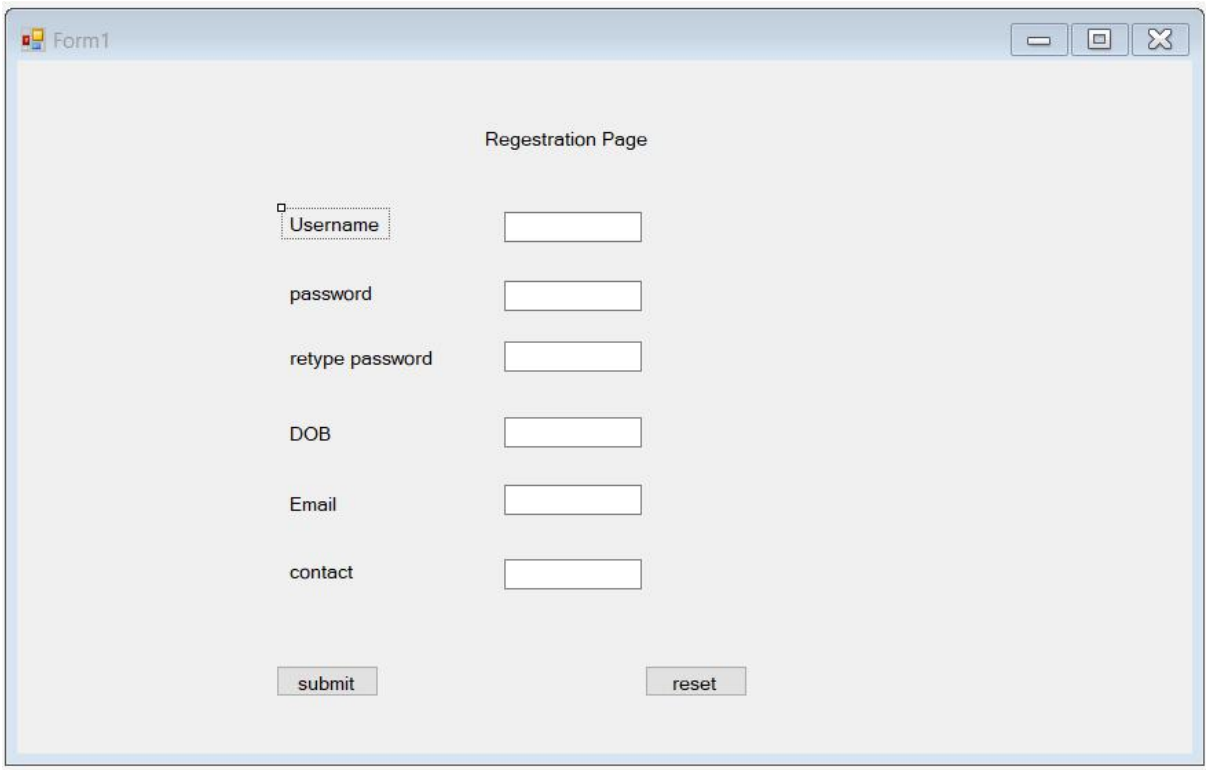


Week 6 (07-10-2020)

LOGIN PAGE - PASSING DATA USING 4 FORMS

1) Create a new project and then select C# Windows Forms Application then click Ok

2) Drag and drop a Label , TextBox and button from the Toolbox. We created a Form with 7 Labels and 6 TextBoxes and 2 buttons
Change the label names and textbox names. we can change the name under "Properties" – Right side corner of application.



3) We have a Username , password , retype password , DOB , Email , Contact Label on the form. So we use three global variables Write the following code in the Form1.cs :

```
public static string SetValueForText1 = "";
public static string SetValueForText2 = "";
public static string SetValueForText3 = "";
public static string SetValueForText1 = "";
public static string SetValueForText2 = "";
public static string SetValueForText3 = "";
```

```
namespace _4_forms
{
    12 references
    public partial class Form1 : Form
    {
        public static string SetValueForText1 = "";
        public static string SetValueForText2 = "";
        public static string SetValueForText3 = "";
        public static string SetValueForText4 = "";
        public static string SetValueForText5 = "";
        public static string SetValueForText6 = "";
        1 reference
        public Form1()
        {
            InitializeComponent();
        }

        1 reference
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
```

4) Add another 2 Windows Forms forms using Project --> Add Windows Form then click on Add(2 times – form2 and form3).

5) After creating the form double-click on the Submit button on the Form1 and write the code:

```
SetValueForText1 = textBox1.Text;
SetValueForText2 = textBox2.Text;
SetValueForText3 = textBox3.Text;
SetValueForText1 = textBox1.Text;
SetValueForText2 = textBox2.Text;
SetValueForText3 = textBox3.Text;

if (Form1.SetValueForText2 == Form1.SetValueForText3)
     {

         Form2 frm2 = new Form2();
         frm2.Show();

         Form3 frm3 = new Form3();
         frm3.Show();
     }
     else
     {
         textBox1.Text = " ";
         textBox2.Text = " ";
         textBox3.Text = " ";
         textBox4.Text = " ";
         textBox5.Text = " ";
         textBox6.Text = " ";
     }
```

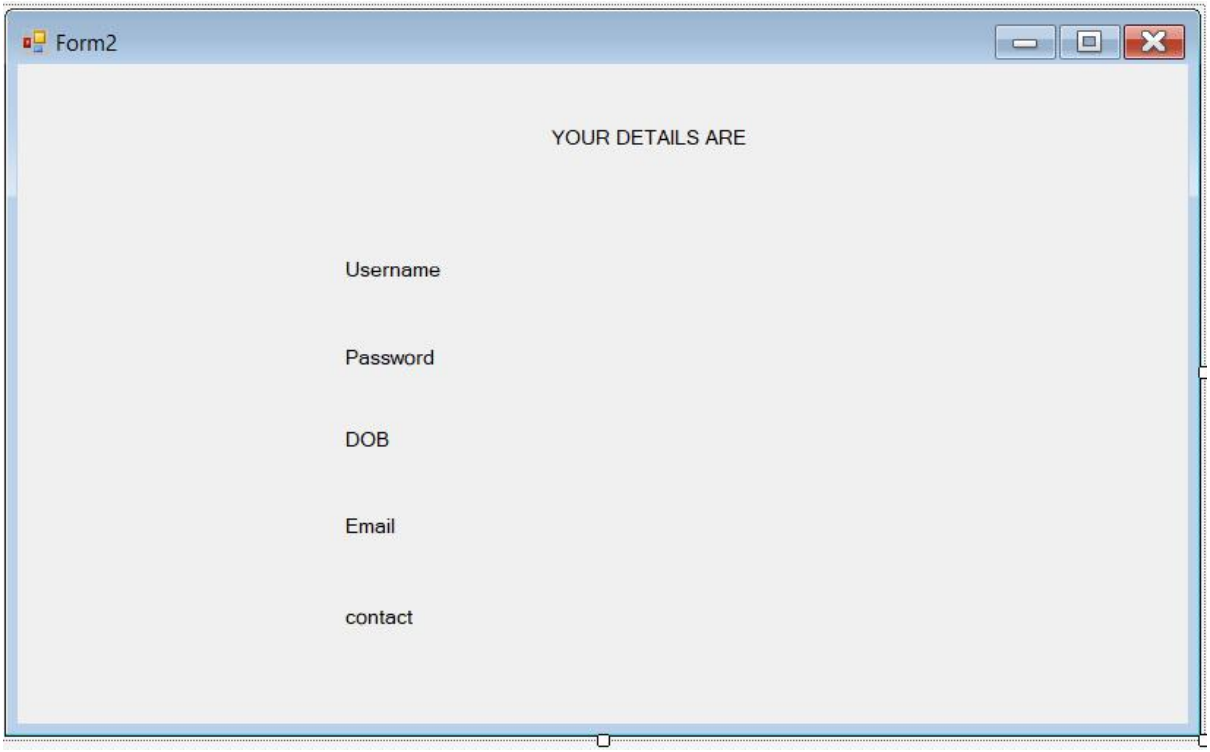
```
private void button1_Click(object sender, EventArgs e)
{
    SetValueForText1 = textBox1.Text;
    SetValueForText2 = textBox2.Text;
    SetValueForText3 = textBox3.Text;
    SetValueForText4 = textBox4.Text;
    SetValueForText5 = textBox5.Text;
    SetValueForText6 = textBox6.Text;

    if (Form1.SetValueForText2 == Form1.SetValueForText3)
    {

        Form2 frm2 = new Form2();
        frm2.Show();

        Form3 frm3 = new Form3();
        frm3.Show();
    }
    else
    {
        textBox1.Text = " ";
        textBox2.Text = " ";
        textBox3.Text = " ";
        textBox4.Text = " ";
        textBox5.Text = " ";
        textBox6.Text = " ";
    }
```

6) Double-click on the Reset button on the login Form and write the code:

```
textBox1.Text = " ";
textBox2.Text = " ";
textBox3.Text = " ";
textBox4.Text = " ";
textBox5.Text = " ";
textBox6.Text = " ";
```

```
1 reference
private void button2_Click(object sender, EventArgs e
{
    textBox1.Text = " ";
    textBox2.Text = " ";
    textBox3.Text = " ";
    textBox4.Text = " ";
    textBox5.Text = " ";
    textBox6.Text = " ";

}
```

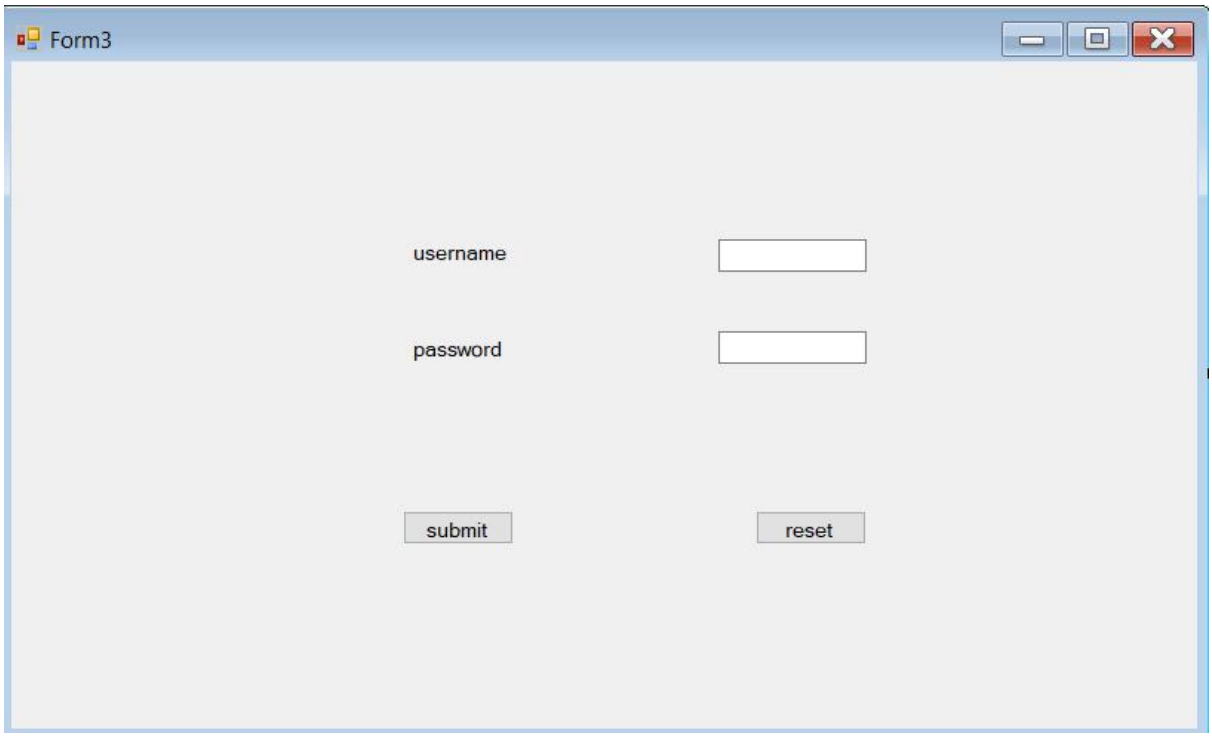
7) Drag and Drop 6 Labels from the Toolbox onto Form2 and rename them.



8) Double-click on Form2 and write the code as below :

```
4 references
public partial class Form2 : Form
{
    1 reference
    public Form2()
    {
        InitializeComponent();
    }

    1 reference
    private void Form2_Load(object sender, EventArgs e)
    {
        label2.Text = Form1.SetValueForText1;
        label3.Text = Form1.SetValueForText2;
        label4.Text = Form1.SetValueForText4;
        label5.Text = Form1.SetValueForText5;
        label6.Text = Form1.SetValueForText6;
    }
}
}
```

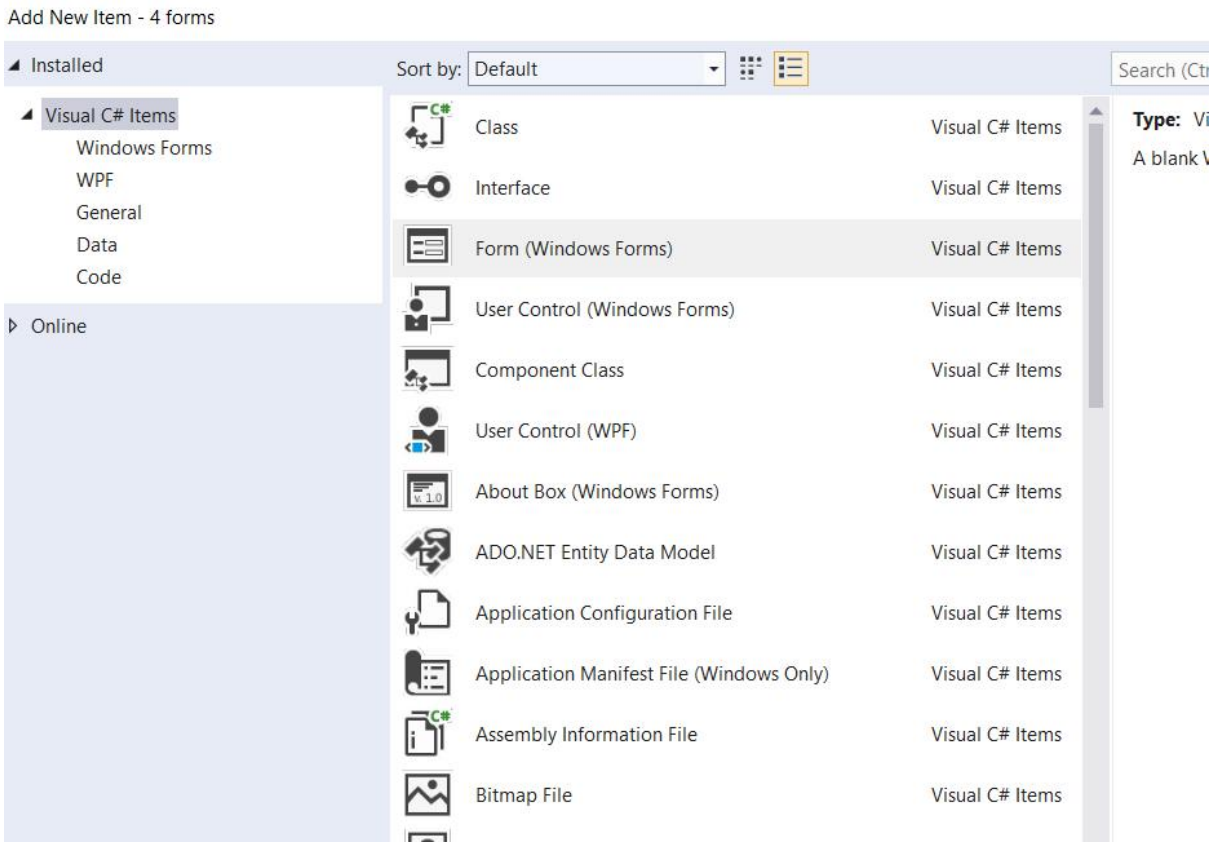
9) Drag and Drop 2 Labels , 2 textboxes and 2 buttons from the Toolbox onto Form2 and rename them.



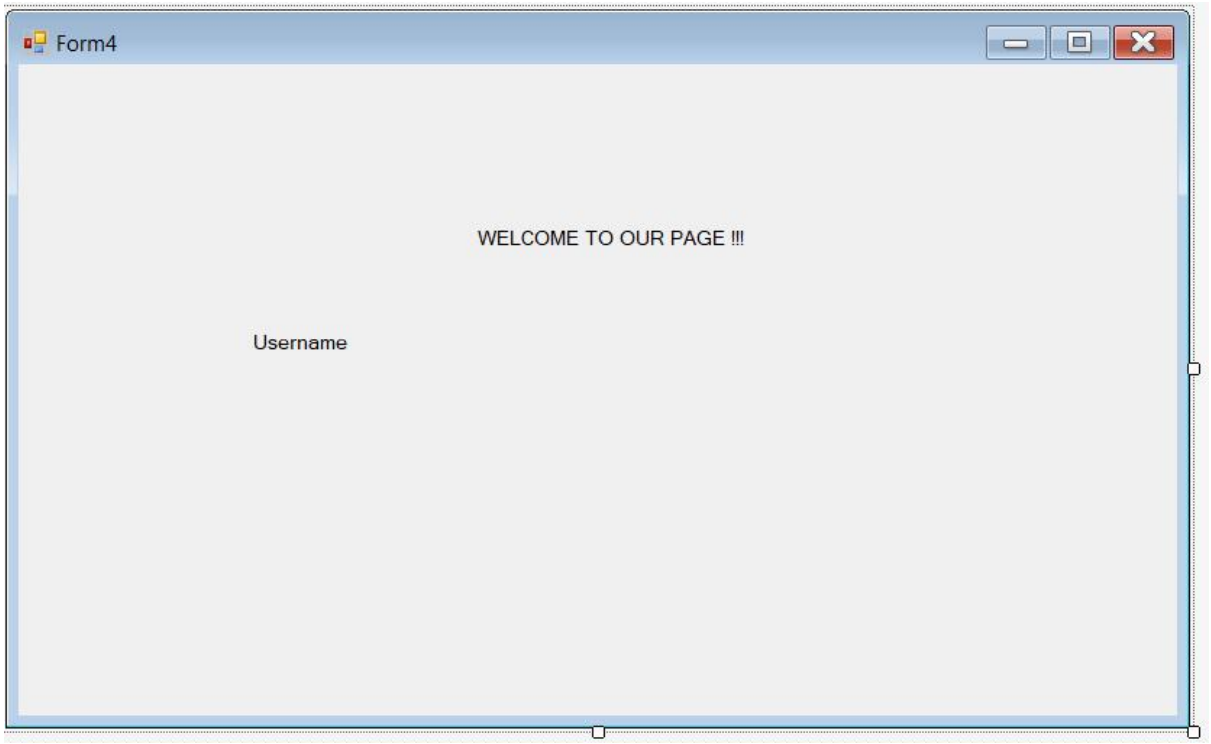
10) We have a Username , password Label on the form. So we use 2 global variables. Write the code in the Form3.cs as shown:

```
public partial class Form3 : Form
{
    public static string SetValueForText1 = "";
    public static string SetValueForText2 = "";
    1 reference
    public Form3()
    {
```

11) Add another Windows Forms form using Project --> Add Windows Form then click on Add.(form 4 is added)



12) Add 2 labels to the form4 and rename them



13) After creating the form4 double-click on the Submit button on the Form3 and write the code:

```
private void button1_Click(object sender, EventArgs e)
{
    SetValueForText1 = textBox1.Text;
    SetValueForText2 = textBox2.Text;

    if(Form3.SetValueForText1==Form1.SetValueForText1 && Form3.SetValueForText2 == Form1.SetValueForText2)
    {

        Form4 frm4 = new Form4();
        frm4.Show();
    }
    else
    {
        textBox1.Text = " ";
        textBox2.Text = " ";
    }
```

14) double-click on the reset button on the Form3 and write the code as below:

```
1 reference
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = " ";
    textBox2.Text = " ";
}
```

15) double click on form4 are write the code as below :

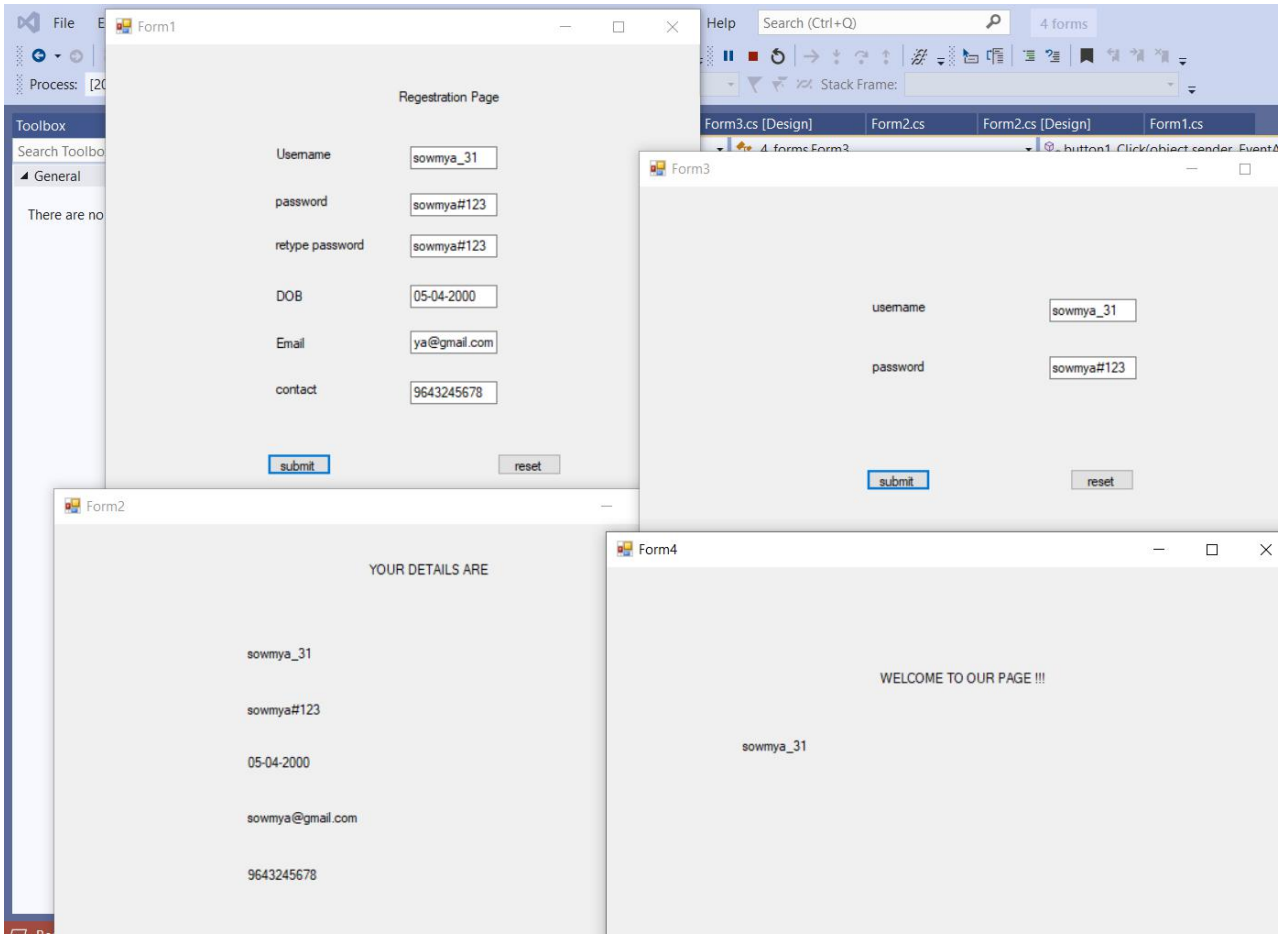
```
    }

    1 reference
    private void Form4_Load(object sender, EventArgs e)
    {
        label2.Text = Form3.SetValueForText1;
    }
}
```

16) Now save and run the application.

Fill in Form1 and click on Submit. if text in password and retype password is same , the data will pass from Form1 (login form) to Form2 and form 3 displayed. else the form gets reset.

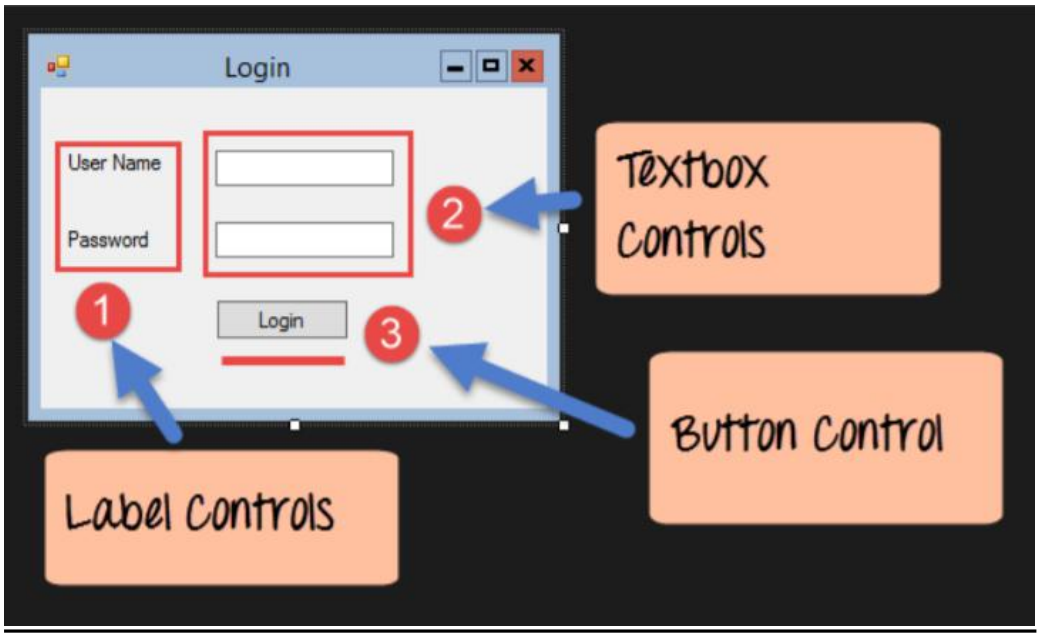
Now, fill the details in form3 and click on submit. If both the username and password are same as that of form1 then form4 is displayed else gets reset.

We can Click on reset button If we want to clear all the details which we've entered in textboxes in form1 or form3.



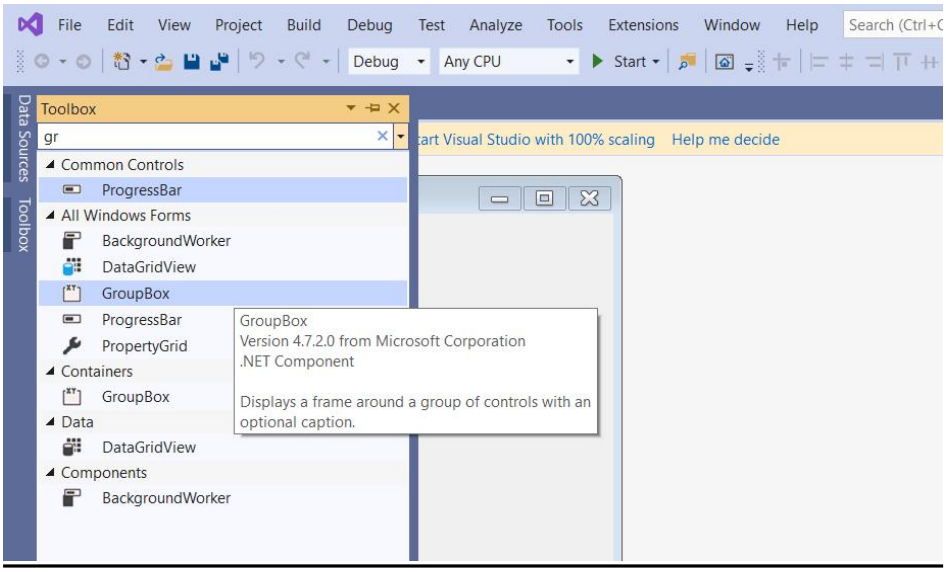
C# Windows Forms Application

A Windows forms application is one that runs on the desktop computer. A Windows forms application will normally have a collection of controls such as labels, textboxes, list boxes, etc.
Below is an example of a simple Windows form application. It shows a simple Login screen, which is accessible by the user. The user will enter the required credentials and then will click the Login button to proceed.

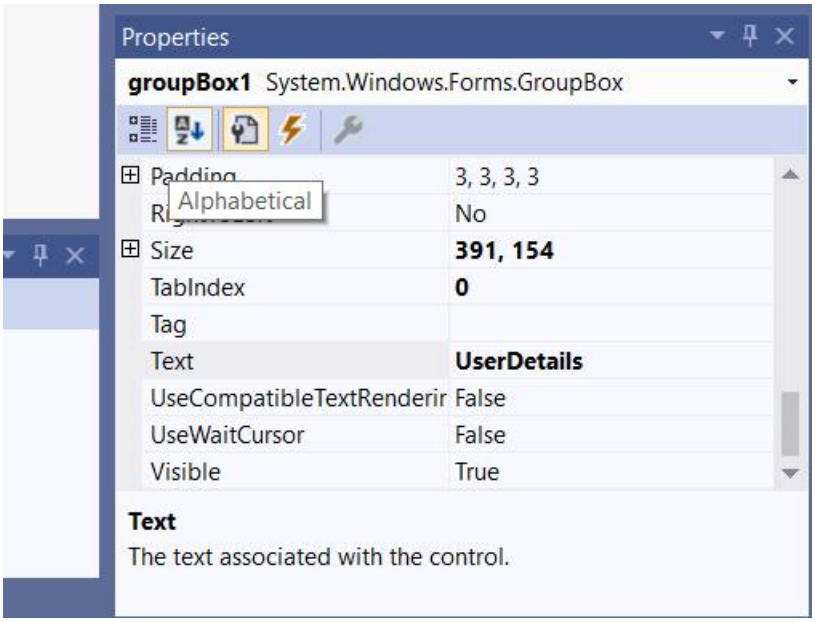


1) Group Box A group box is used for logical grouping controls into a section.

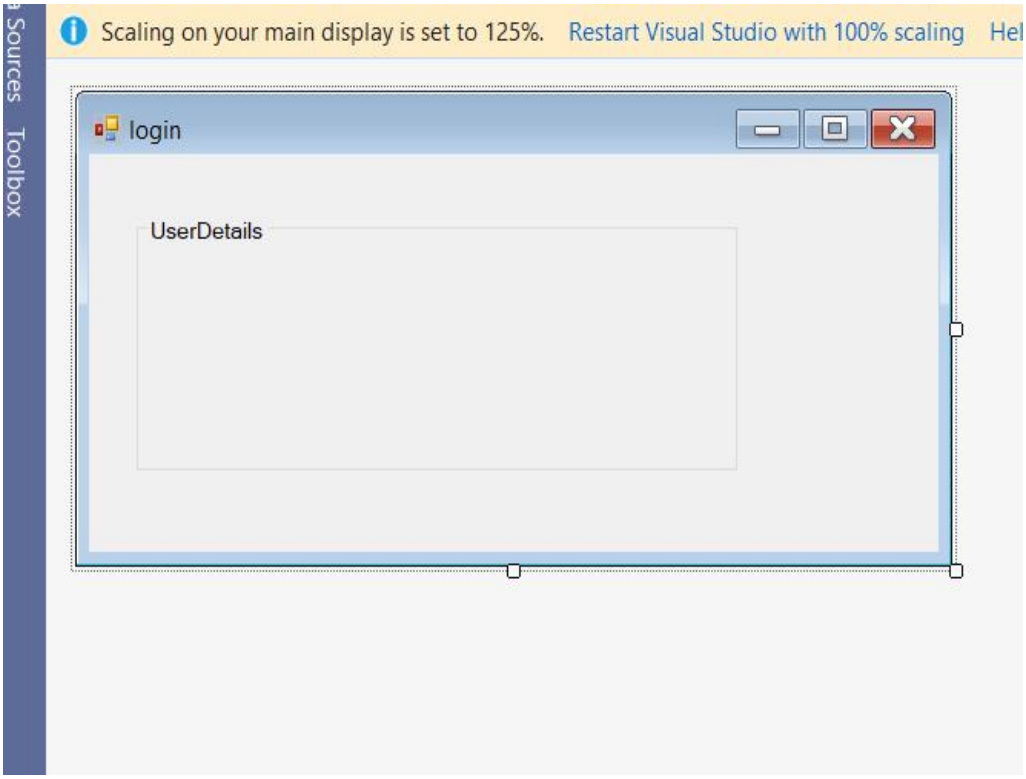
   Step 1) The first step is to drag the Group box control onto the Windows Form from the toolbox as shown below



Step 2) Once the groupbox has been added, go to the properties window by clicking on the groupbox control. In the properties window, go to the Text property and change it to "User Details".



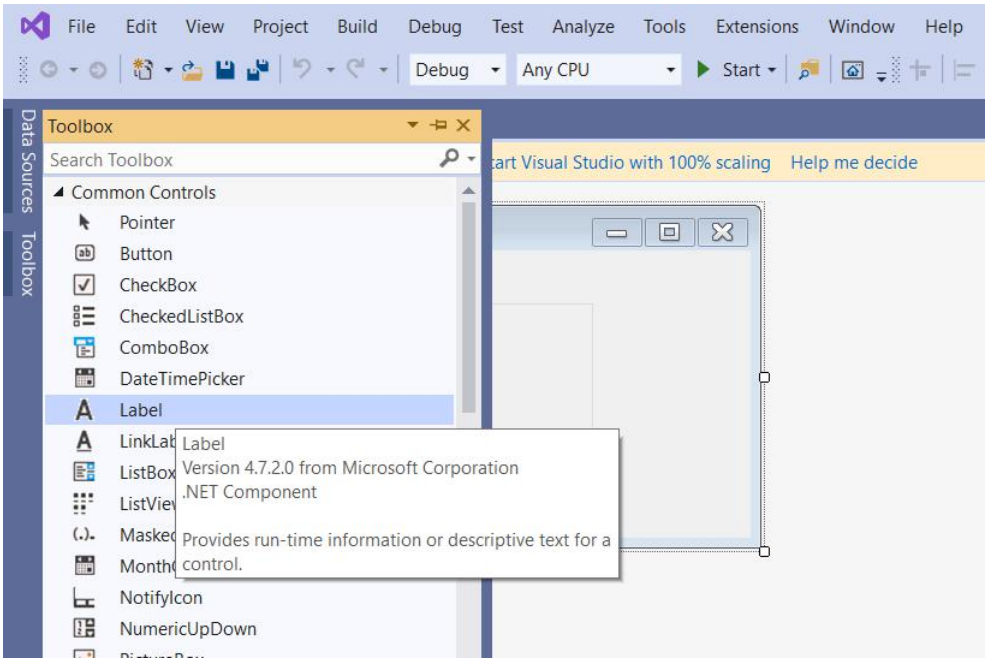
Once the changes are made, the output is :



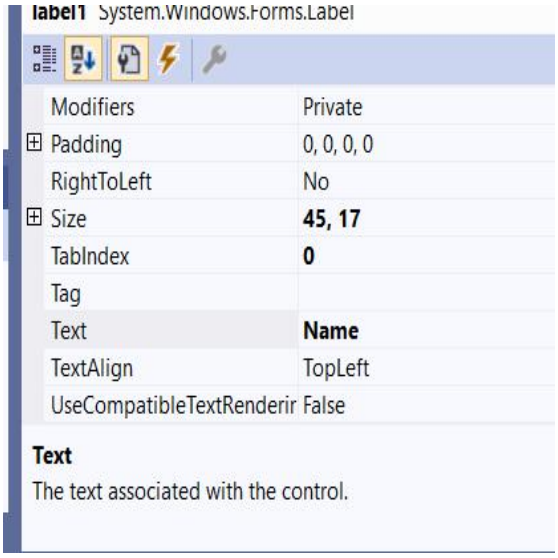
In the output, we can clearly see that the Groupbox was added to the form. we can also see that the text of the groupbox was changed to "User Details."

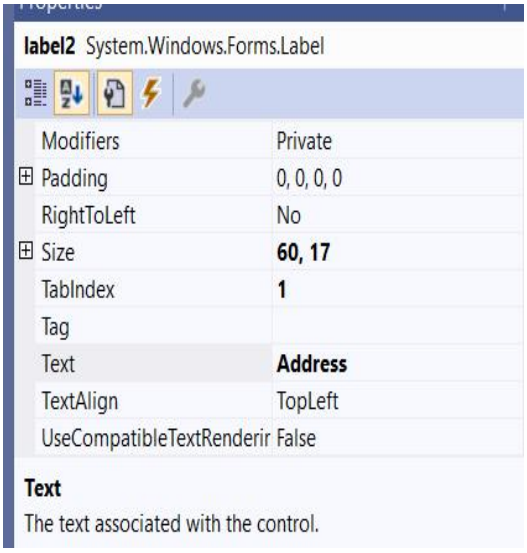
2) Label Control The label control is used to display a text or a message to the user on the form.

Step 1) The first step is to drag the label control on to the Windows Form from the toolbox as shown below. Make sure to drag the label control 2 times so that you can have one for the ‘name’ and the other for the ‘address’.

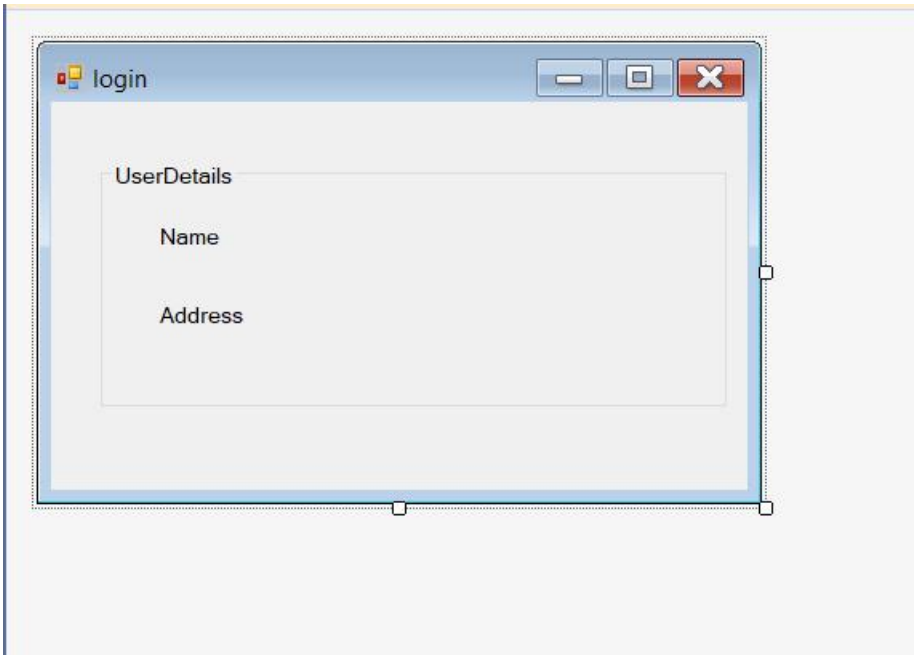


Step 2) Once the labels have been added, go to the properties window by clicking on the label control. In the properties window, go to the Text property of each label control and change them.



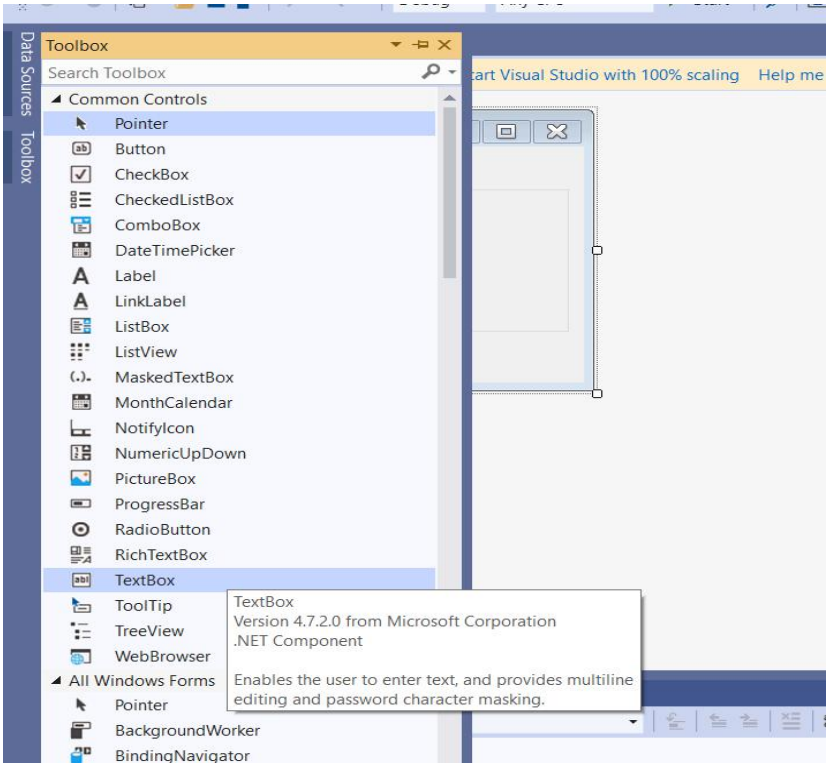


Once the changes are made, we will see the following output:

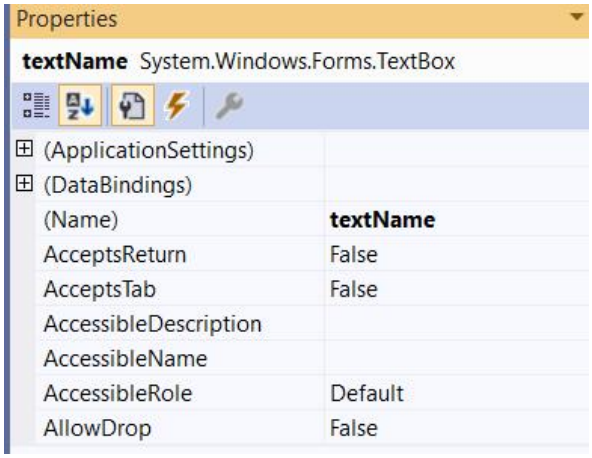


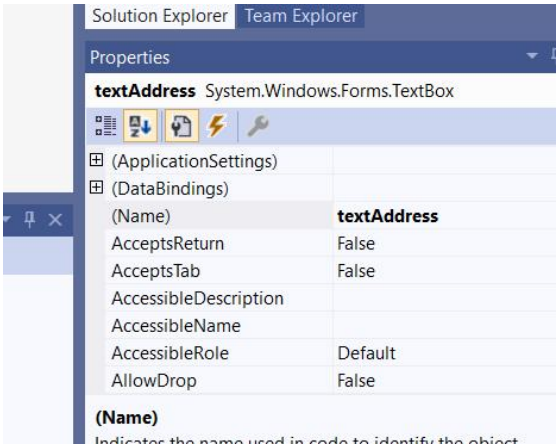
3) TextBox: A textbox is used for allowing a user to enter some text on the forms application.

Step 1) The first step is to drag the textbox control onto the Windows Form from the toolbox as shown below.

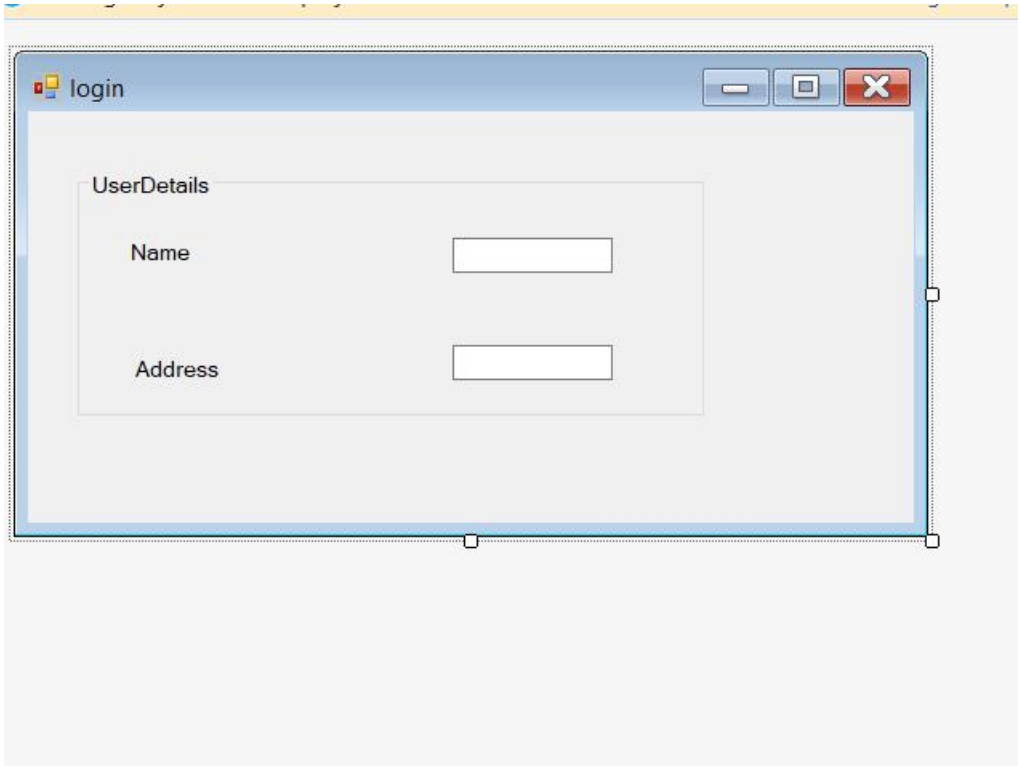


Step 2) Once the text boxes have been added, go to the properties window by clicking on the textbox control. In the properties window, go to the Name property and add a meaningful name to each textbox. For example, name the textbox for the user as textName and that for the address as textAddress. A naming convention and standard should be made for controls because it becomes easier to add extra functionality to these controls.



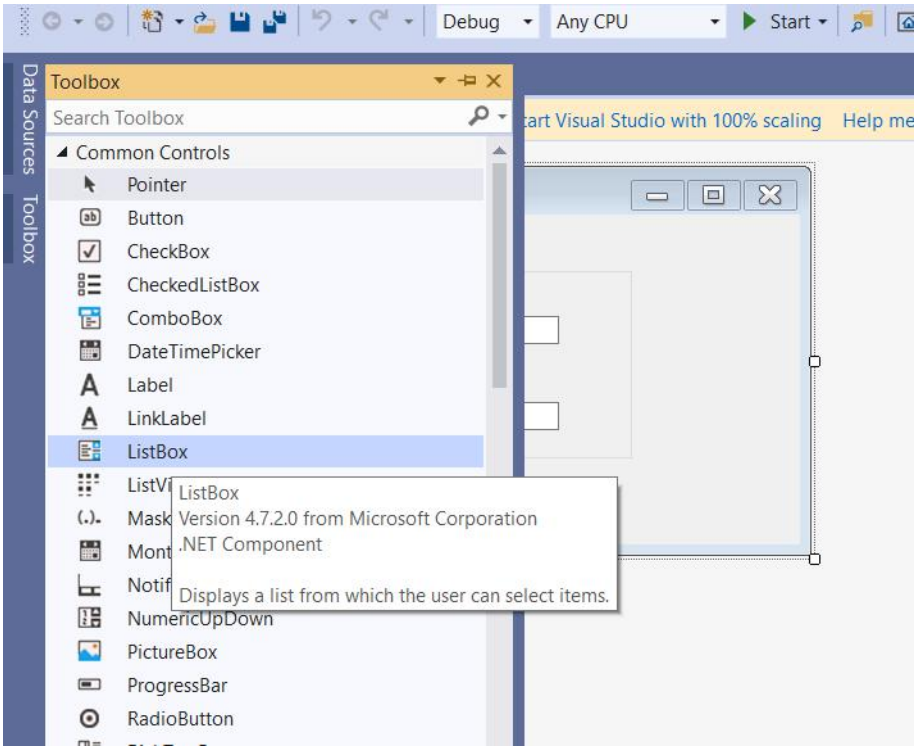


Once you make the above changes, you will see the following output:



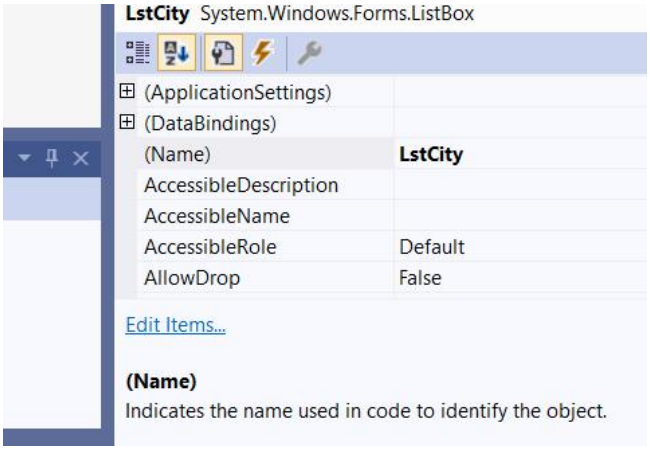
4) List box: A List box is used to showcase a list of items on the Windows form.

Step 1) The first step is to drag the list box control onto the Windows Form from the toolbox as shown below

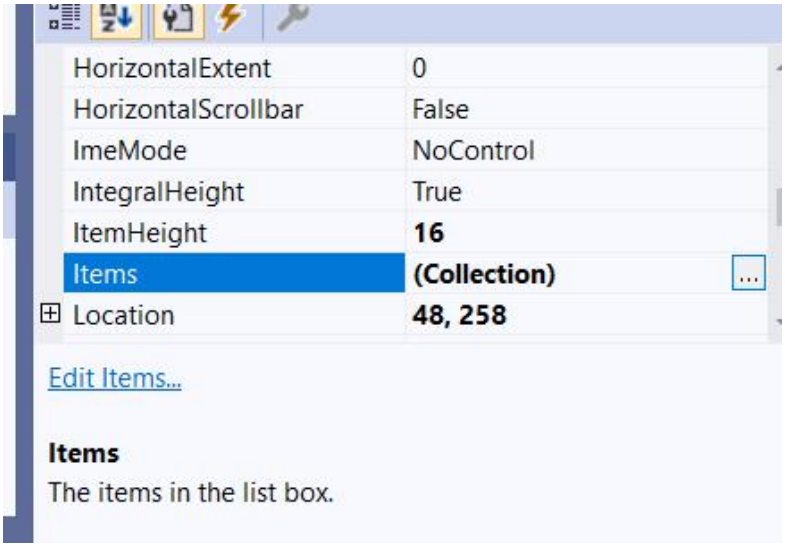


Step 2) Once the list box has been added, go to the properties window by clicking on the list box control.

1. First, change the property of the Listbox box control, in our case, we have changed this to lstCity

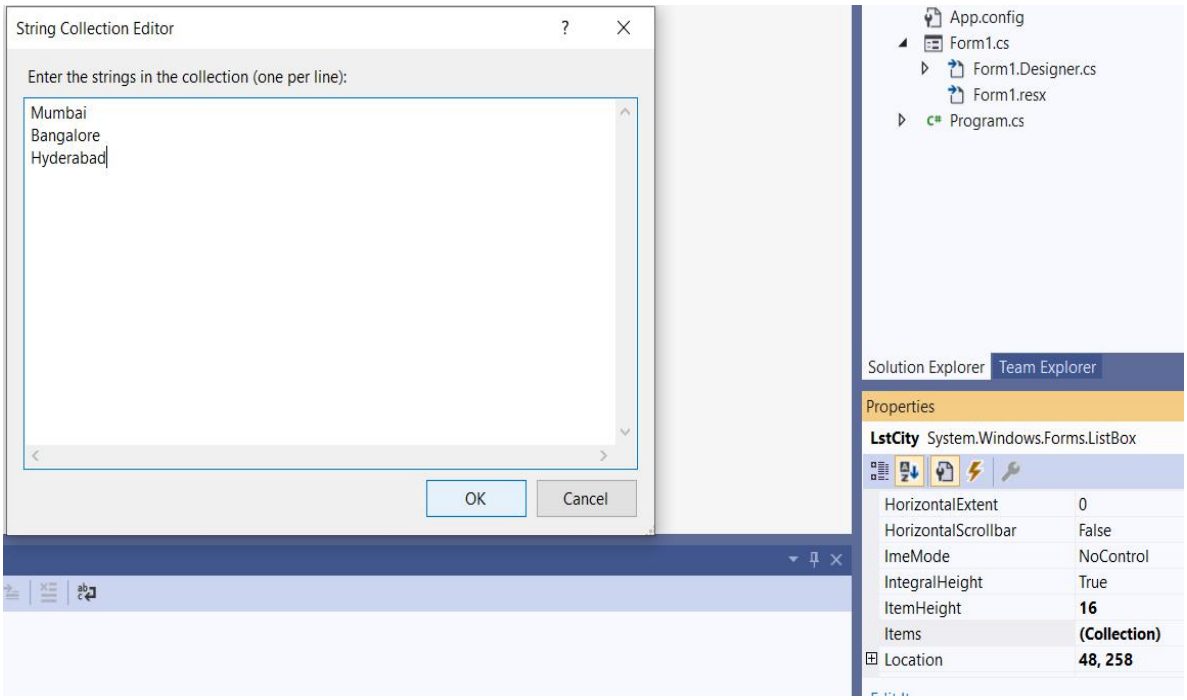


2. Click on the Items property. This will allow you to add different items which can show up in the list box. In our case, we have selected items "collection".

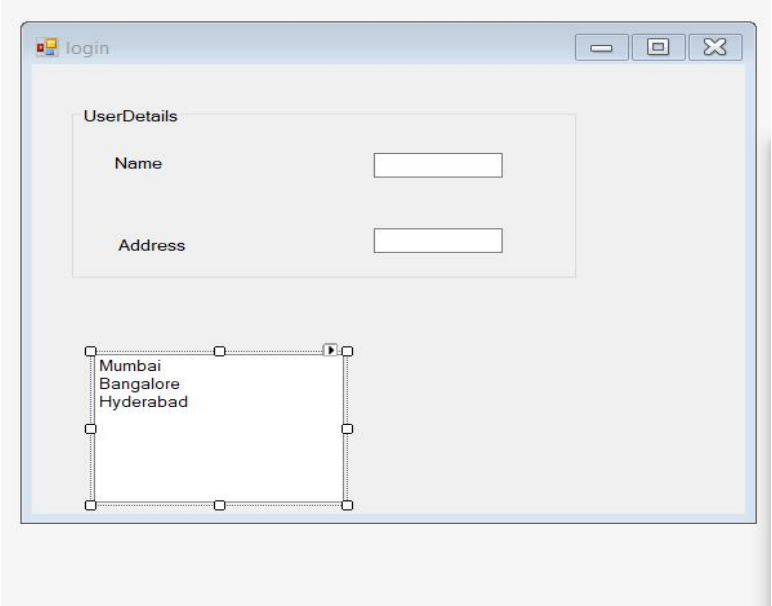


3. In the String Collection Editor, which pops up, enter the city names. In our case, we have entered "Mumbai", "Bangalore" and "Hyderabad".

4. Finally, click on the 'OK' button.

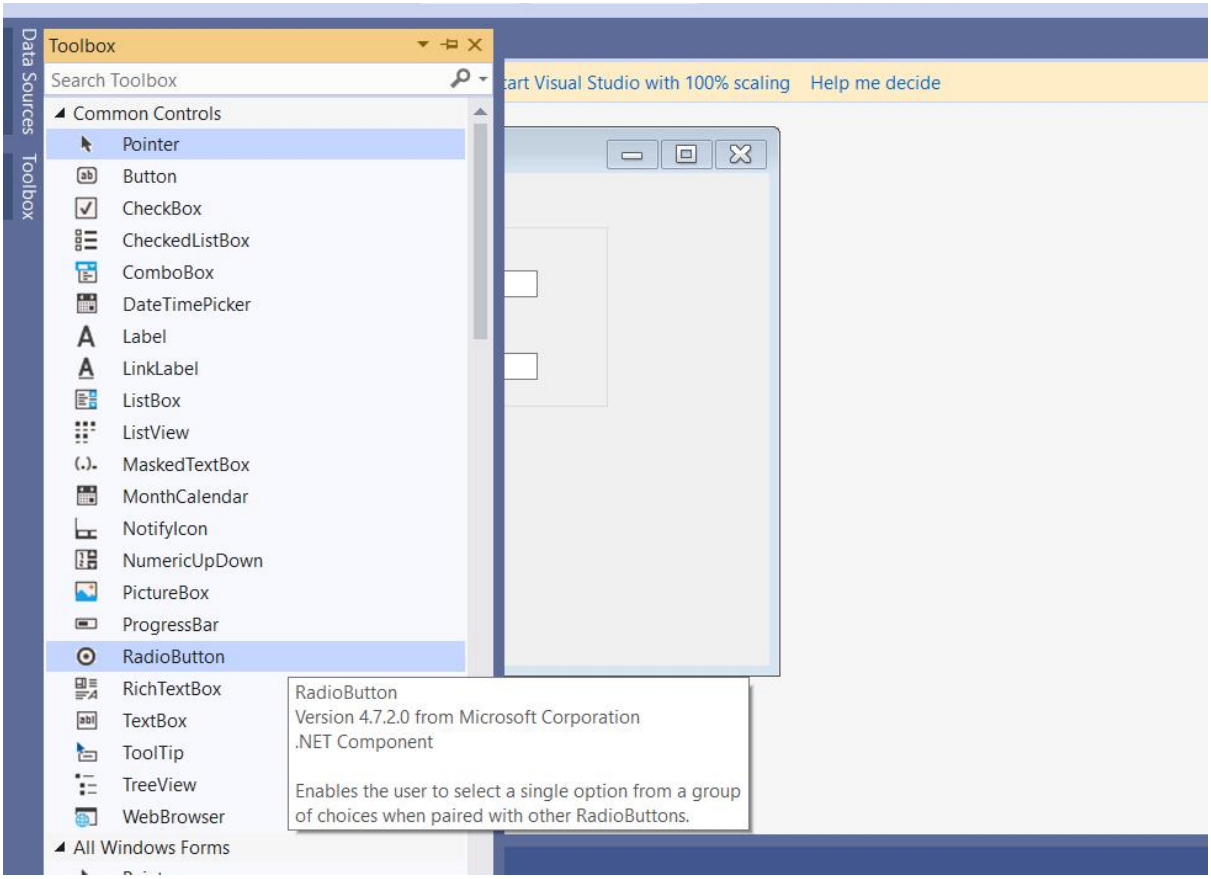


Once the changes are made, we will see the following output:



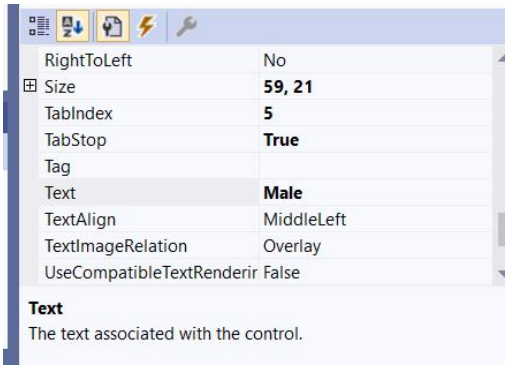
5) RadioButton :A Radiobutton is used to showcase a list of items out of which the user can choose one.

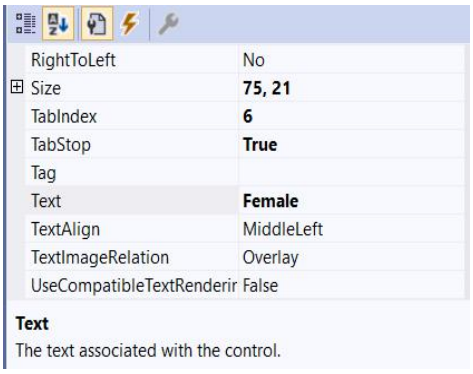
Step 1) The first step is to drag the 'radiobutton' control onto the Windows Form from the toolbox as shown below.



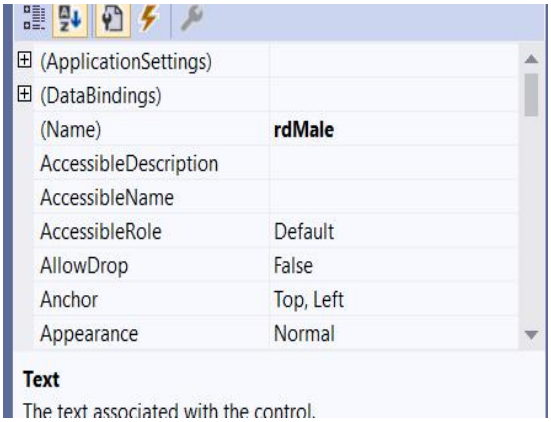
Step 2) Once the Radiobutton has been added, go to the properties window by clicking on the Radiobutton control.

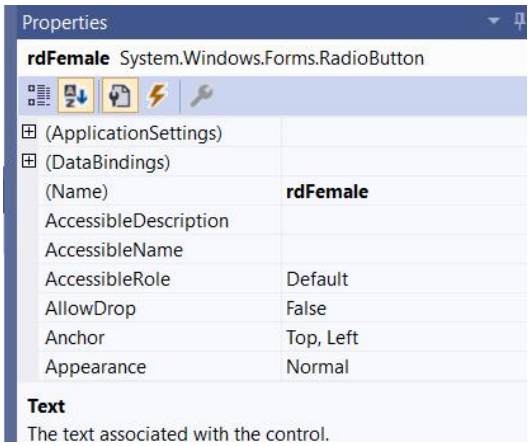
1. First, you need to change the text property of both Radio controls. Go the properties windows and change the text to a male of one radiobutton and the text of the other to female.



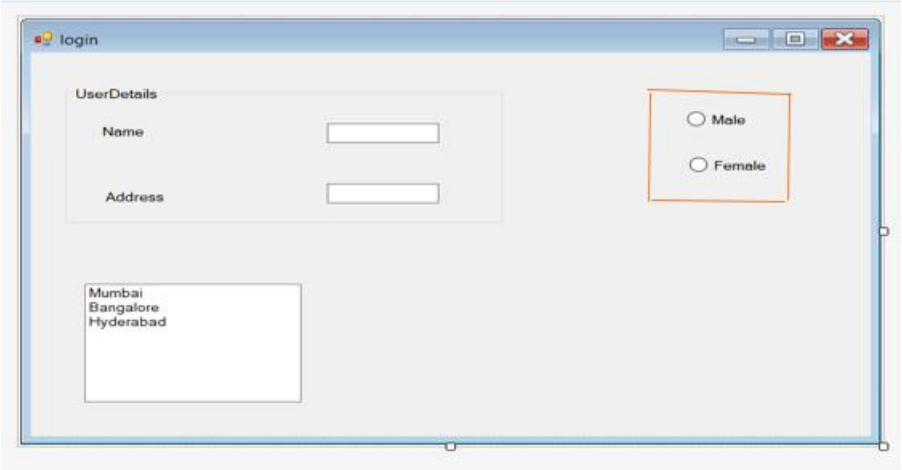


2. Similarly, change the name property of both Radio controls. Go the properties windows and change the name to 'rdMale' of one radiobutton and to 'rdfemale' for the other one.



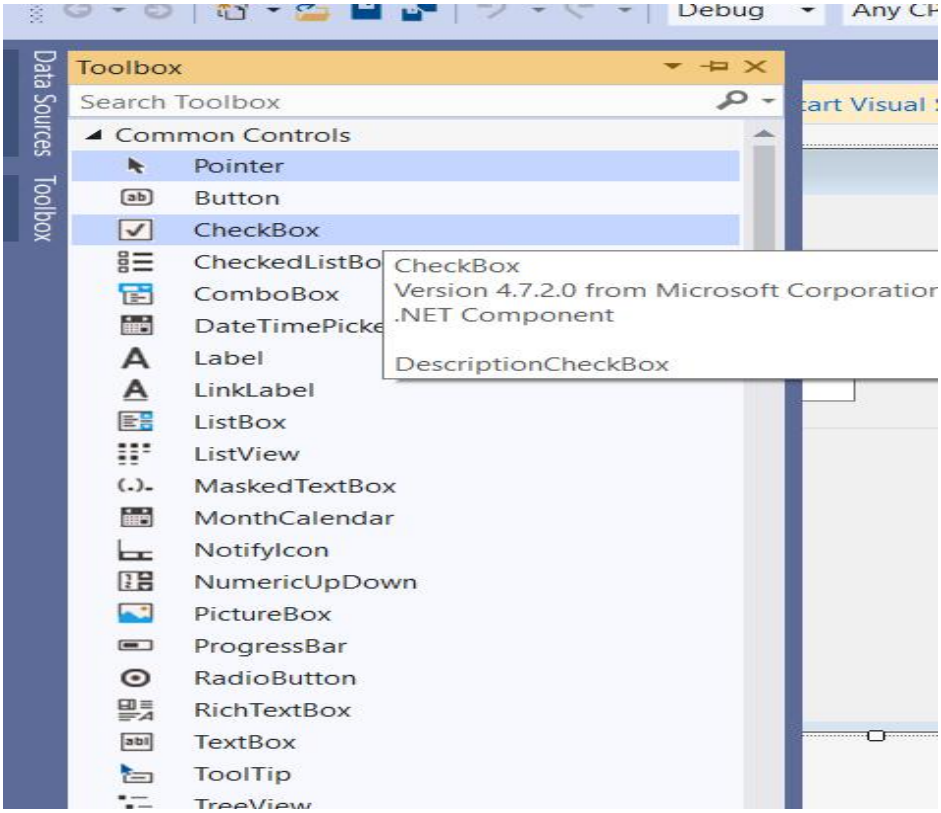


One the changes are made, we will see the following output



6) Checkbox :A checkbox is used to provide a list of options in which the user can choose multiple choices.

Step 1) The first step is to drag the checkbox control onto the Windows Form from the toolbox as shown below

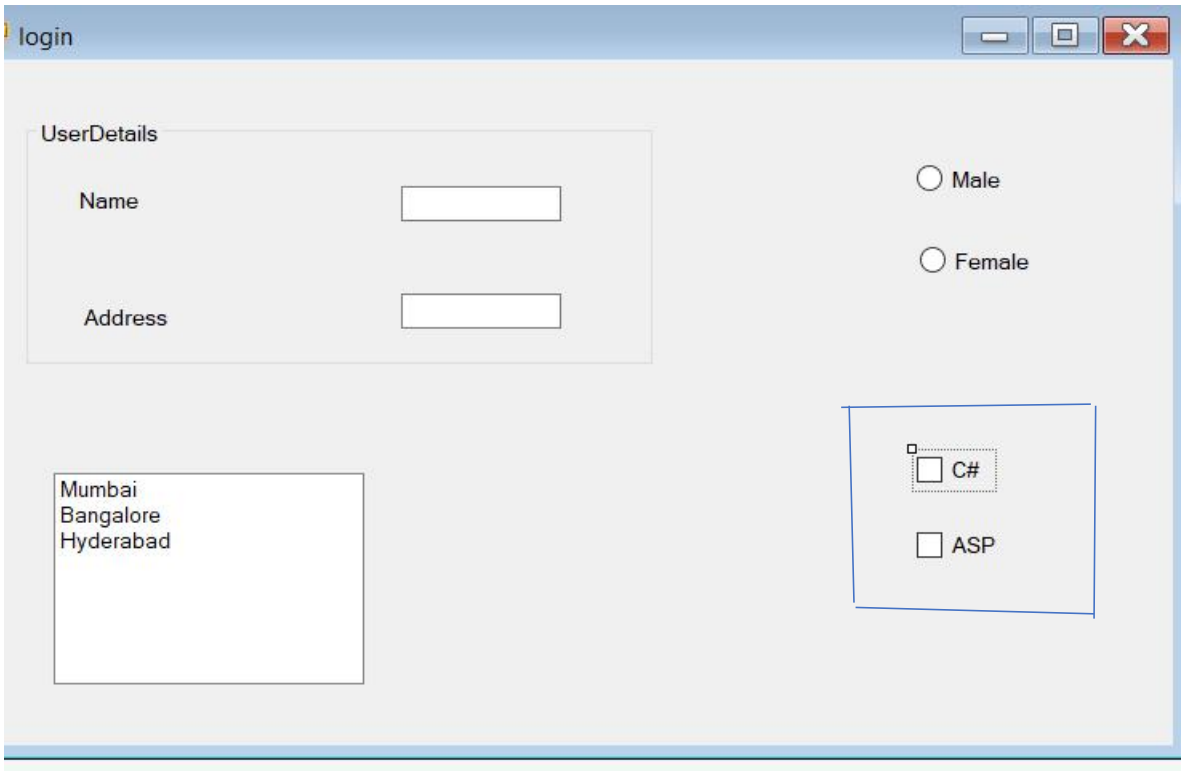


Step 2) Once the checkbox has been added, go to the properties window by clicking on the Checkbox control.

In the properties window,

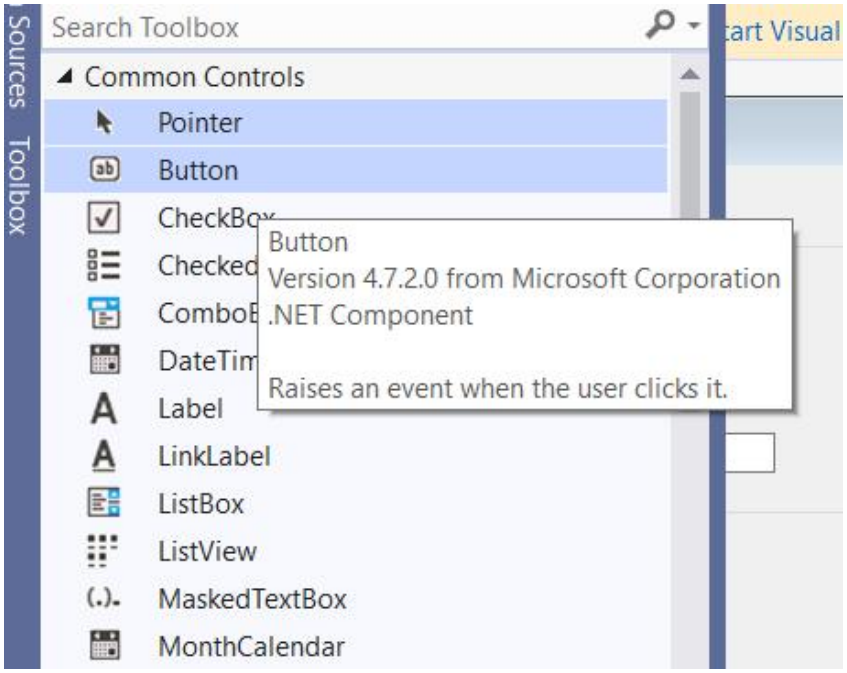
1. First, you need to change the text property of both checkbox controls. Go the properties
   windows and change the text to C# and ASP.Net.
2. Similarly, change the name property of both Radio controls. Go the properties windows
   and change the name to chkC of one checkbox and to chkASP for the other one.

Once we make the above changes, we will see the following output:



7) Button :A button is used to allow the user to click on a button which would then start the processing of the form

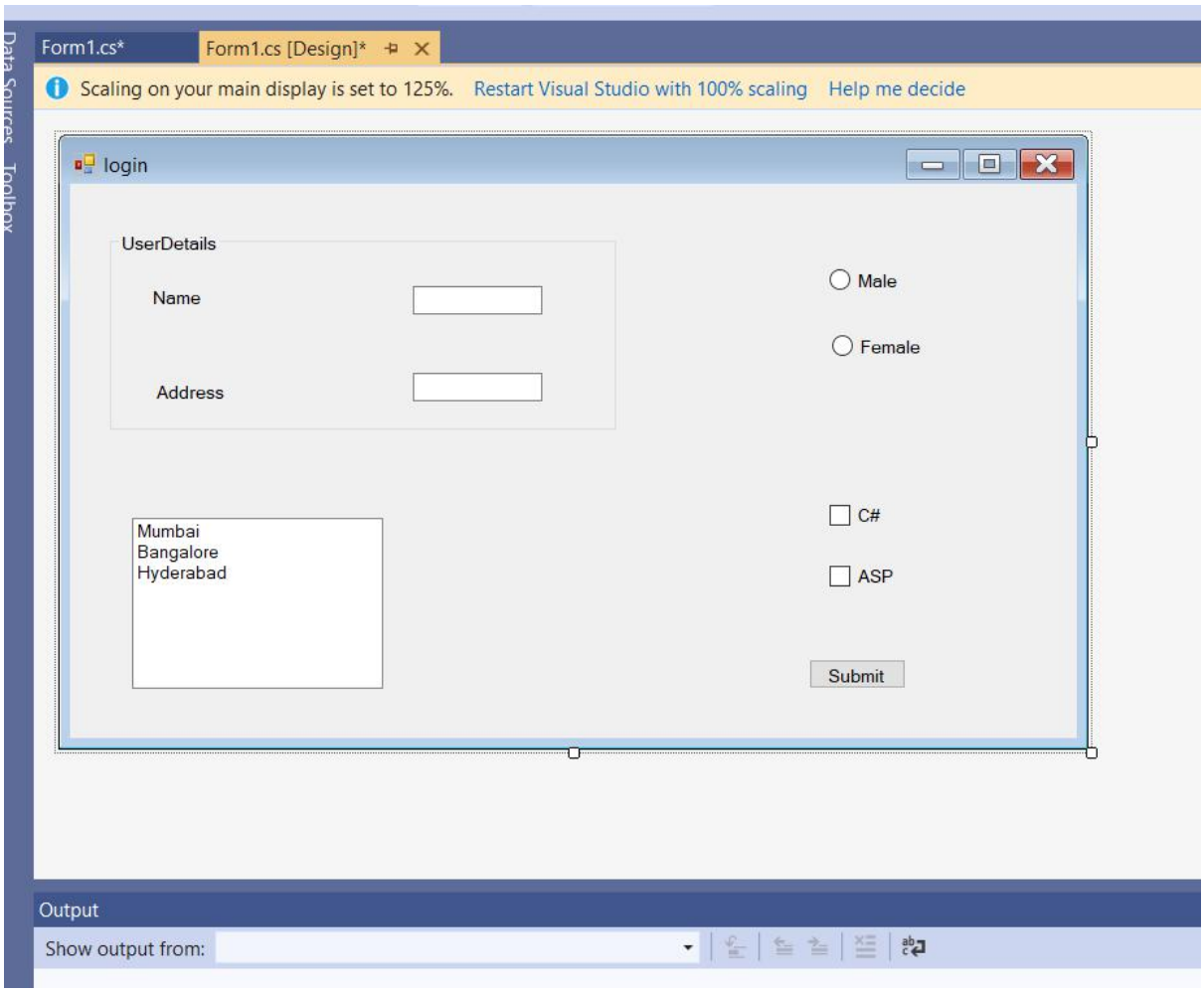
Step 1) The first step is to drag the button control onto the Windows Form from the toolbox as shown below



Step 2) Once the Button has been added, go to the properties window by clicking on the Button control.

1. First, you need to change the text property of the button control. Go the properties windows and change the text to 'submit'.
2. Similarly, change the name property of the control. Go the properties windows and change the name to 'btnSubmit'.

Once you make the above changes, you will see the following output:



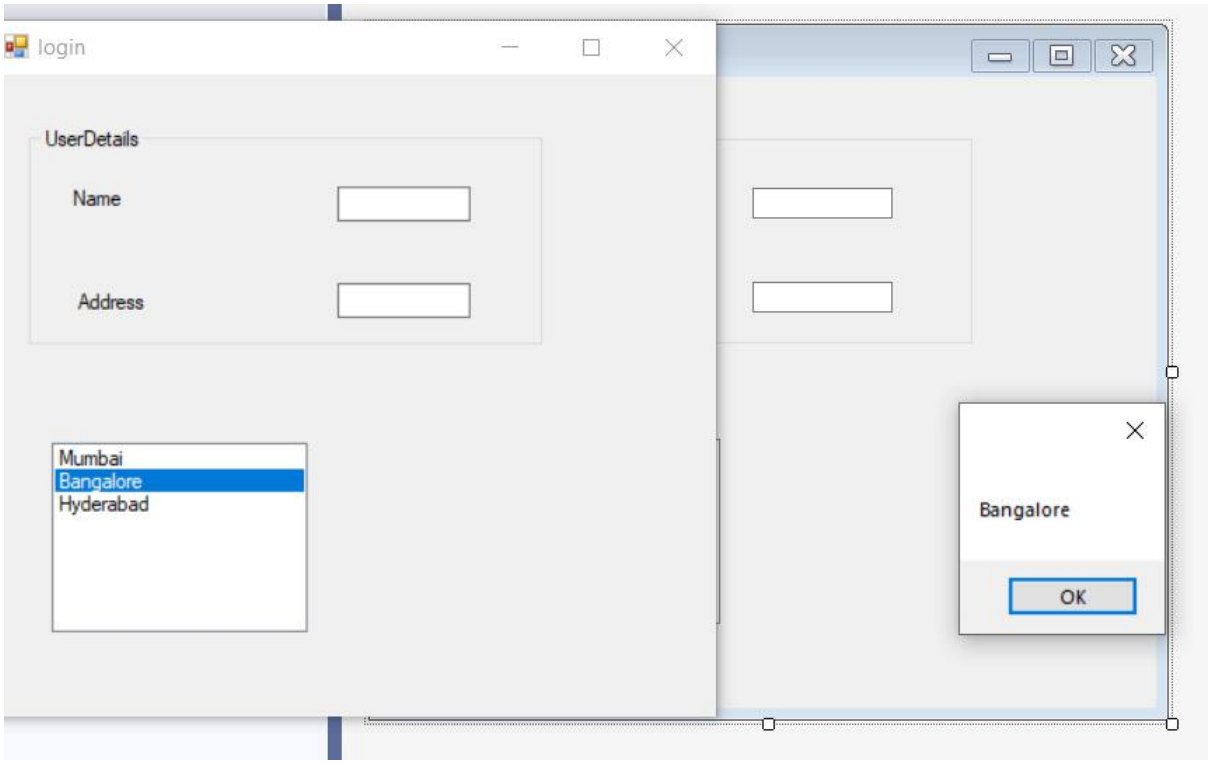
Double click on the Listbox in the form designer. By doing this, Visual Studio will automatically open up the code file for the form. And it will automatically add an event method to the code. This event method will be triggered, whenever any item in the listbox is selected.

Now let's add the below section of code to this snippet of code, to add the required functionality to the listbox event:

```
string text = LstCity.GetItemText(LstCity.SelectedItem);
MessageBox.Show(text);
```

```
1 reference
private void LstCity_SelectedIndexChanged(object sender, EventArgs e)
{
    string text = LstCity.GetItemText(LstCity.SelectedItem);
    MessageBox.Show(text);
}
```

1. This is the event handler method which is automatically created by Visual Studio when you double-click the List box control. You don't need to worry about the complexity of the method name or the parameters passed to the method.
2. Here we are getting the SelectedItem through the lstCity.SelectedItem property. Remember that lstCity is the name of our Listbox control. We then use the GetItemText method to get the actual value of the selected item. We then assign this value to the text variable.
3. Finally, we use the MessageBox method to display the text variable value to the user.

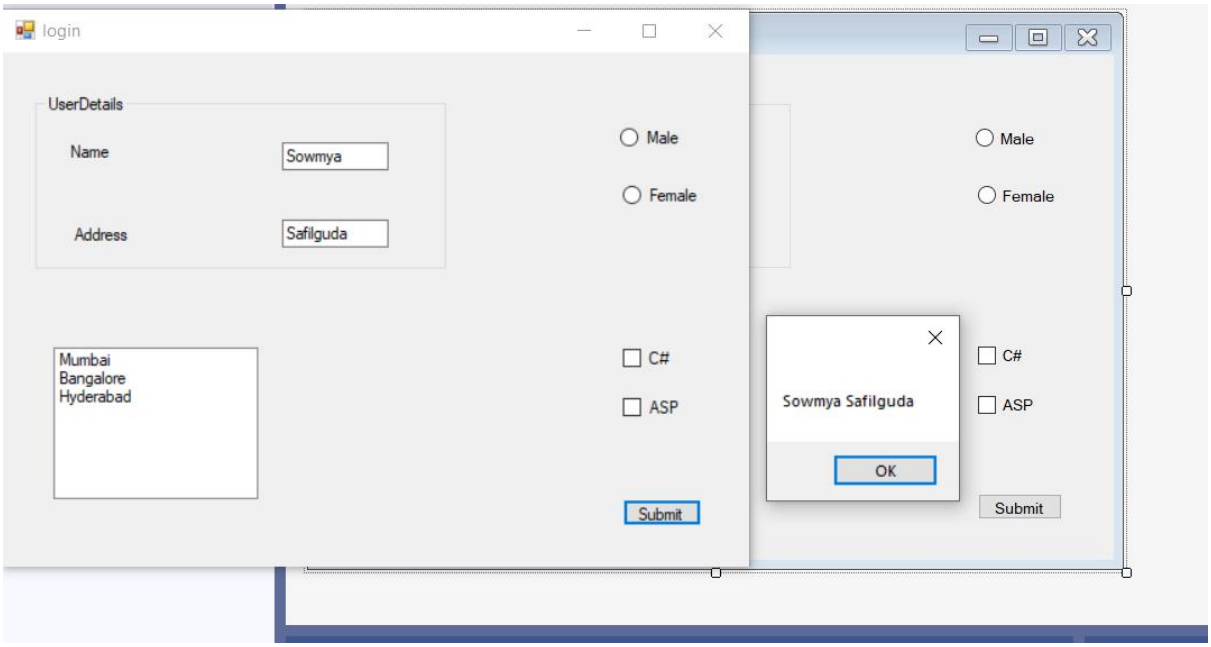


From the output, you can see that when any item from the list box is selected, a message box will pops up. This will show the selected item from the listbox.

Now let's look at the final control which is the button click Method. Again this follows the same philosophy. Just double click the button in the Forms Designer and it will automatically add the method for the button event handler. Then you just need to add the below code.

```
1 reference
private void btnSubmit_Click(object sender, EventArgs e)
{
    string name = textName.Text;
    string address = textAddress.Text;
    MessageBox.Show(name + address);
}

0 references
```

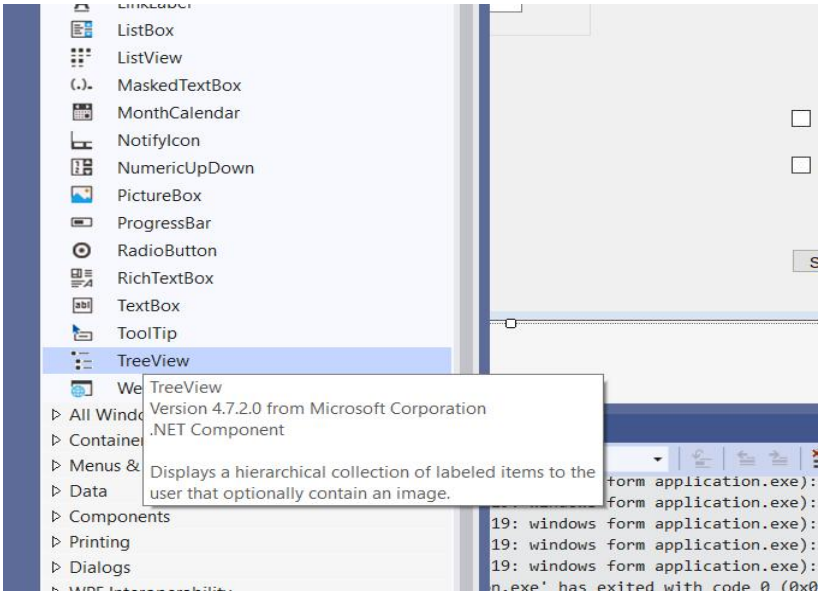
1. This is the event handler method which is automatically created by Visual Studio when you double click the button control. You don't need to worry on the complexity of the method name or the parameters passed to the method.
2. Here we are getting values entered in the name and address textbox. The values can be taken from the text property of the textbox. We then assign the values to 2 variables, name, and address accordingly.
3. Finally, we use the MessageBox method to display the name and address values to the user.



8) Tree and PictureBox Control: There are 2 further controls we can look at, one is the 'Tree Control' and the other is the 'Image control'. Let's look at examples of how we can implement these controls :

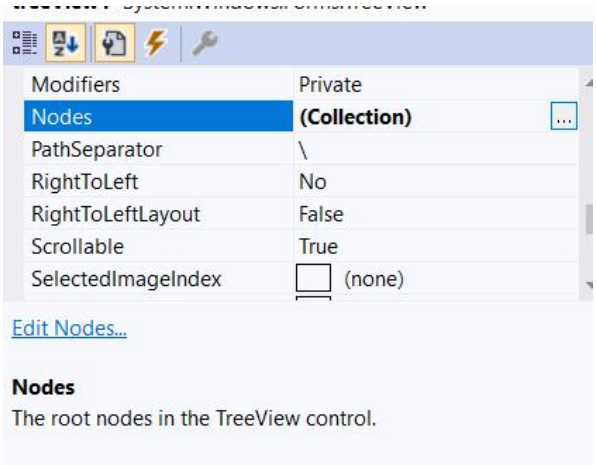
The tree control is used to list down items in a tree like fashion. Probably the best example is when we see the Windows Explorer itself. The folder structure in Windows Explorer is like a tree-like structure. Let's see how we can implement this with an example shown below.

Step 1) The first step is to drag the Tree control onto the Windows Form from the toolbox as shown below:

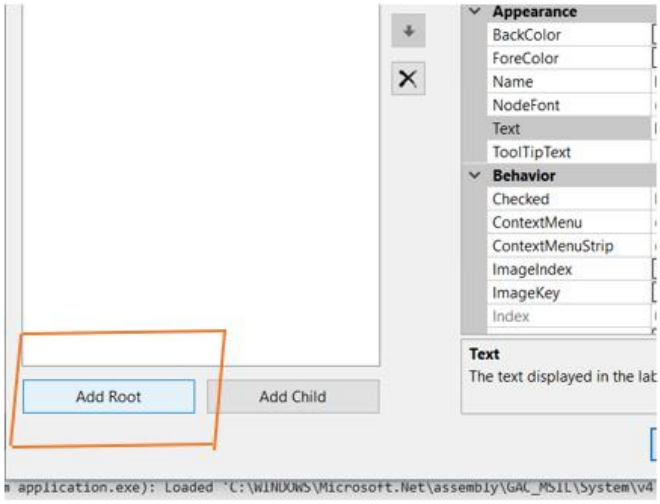


Step 2) The next step is to start adding nodes to the tree collection so that it can come up in the tree accordingly. First, let's follow the below sub-steps to add a root node to the tree collection.

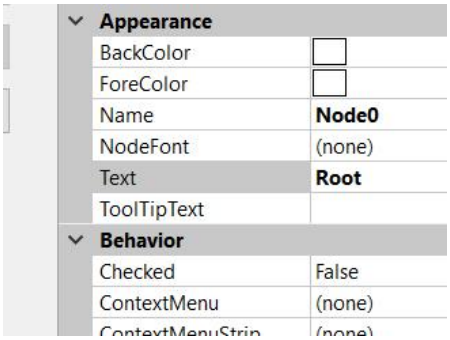
1. Go to the properties toolbox for the tree view control. Click on the Node's property. This will bring up the TreeNode Editor



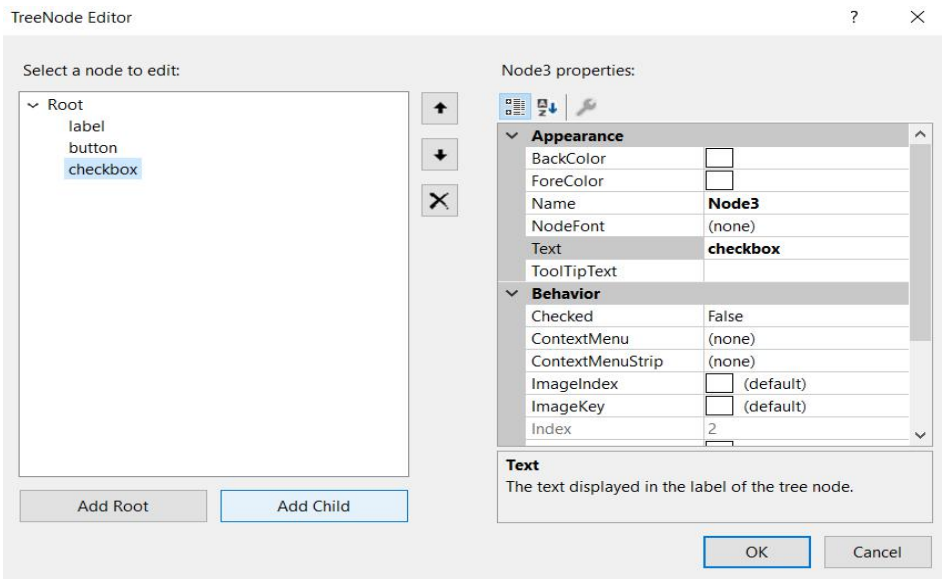
2. In the TreeNode Editor click on the Add Root button to add a root node to the tree collection.



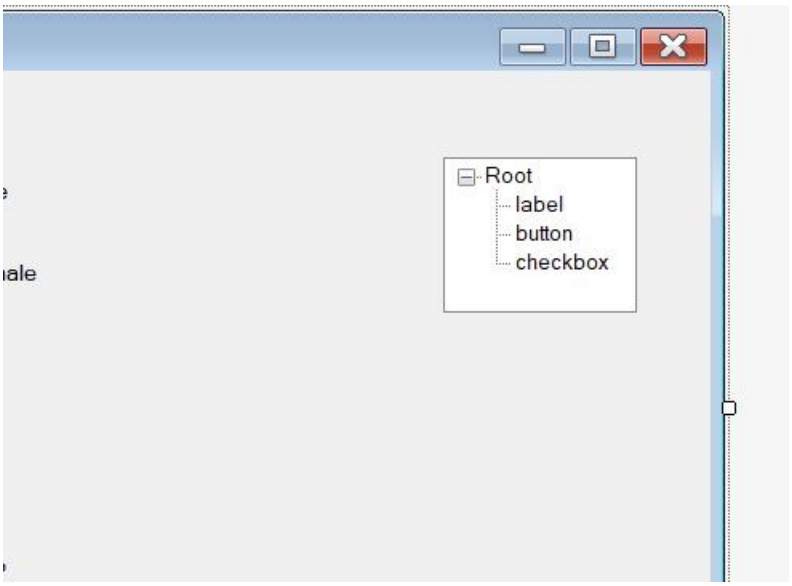
3. Next, change the text of the Root node and provide the text as Root and click 'OK' button. This will add Root node



Step 3) The next step is to start adding the child nodes to the tree collection. Let's follow the below sub-steps to add child root node to the tree collection.

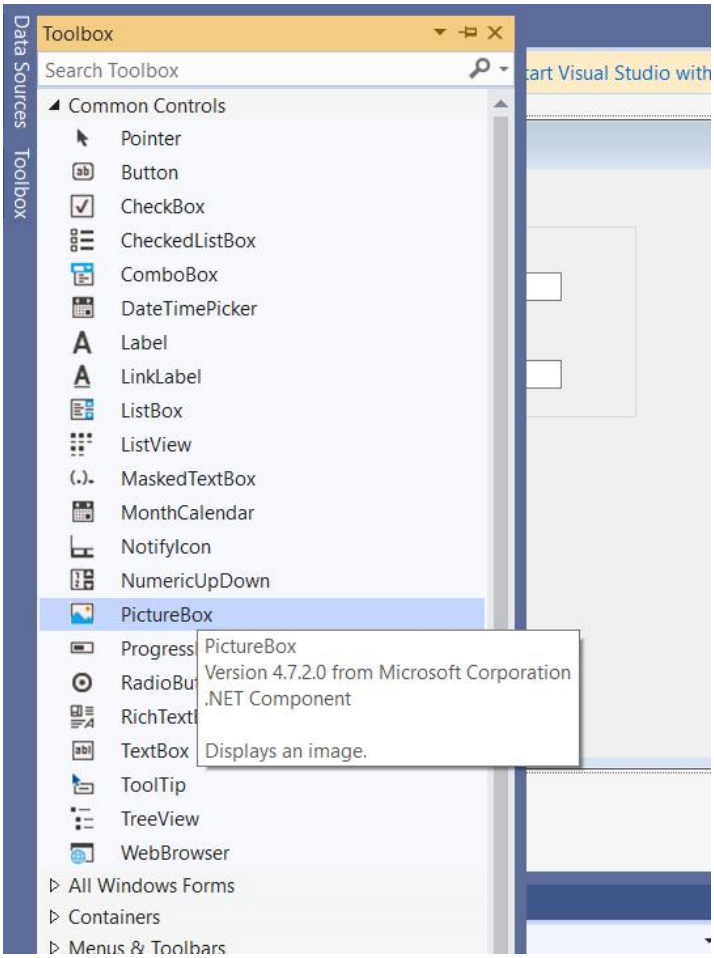


Once the changes are made, we will see the following output:



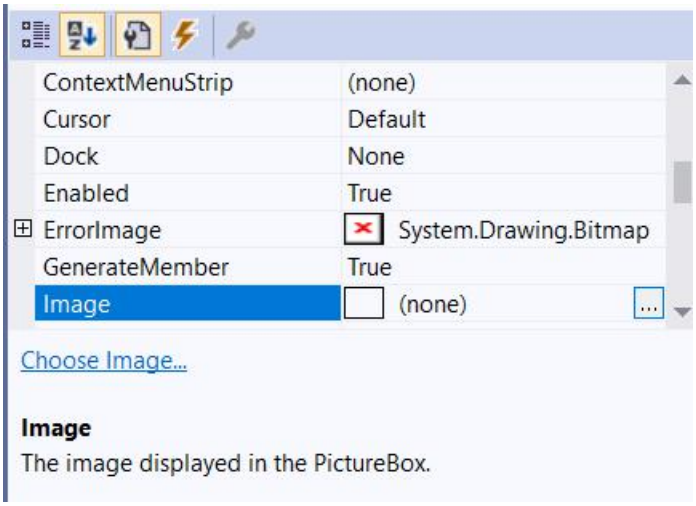
PictureBox Control: This control is used to add images to the Windows Forms. Let's see how we can implement this with an example shown below.

Step 1) The first step is to drag the PictureBox control onto the Windows Form from the toolbox as shown below

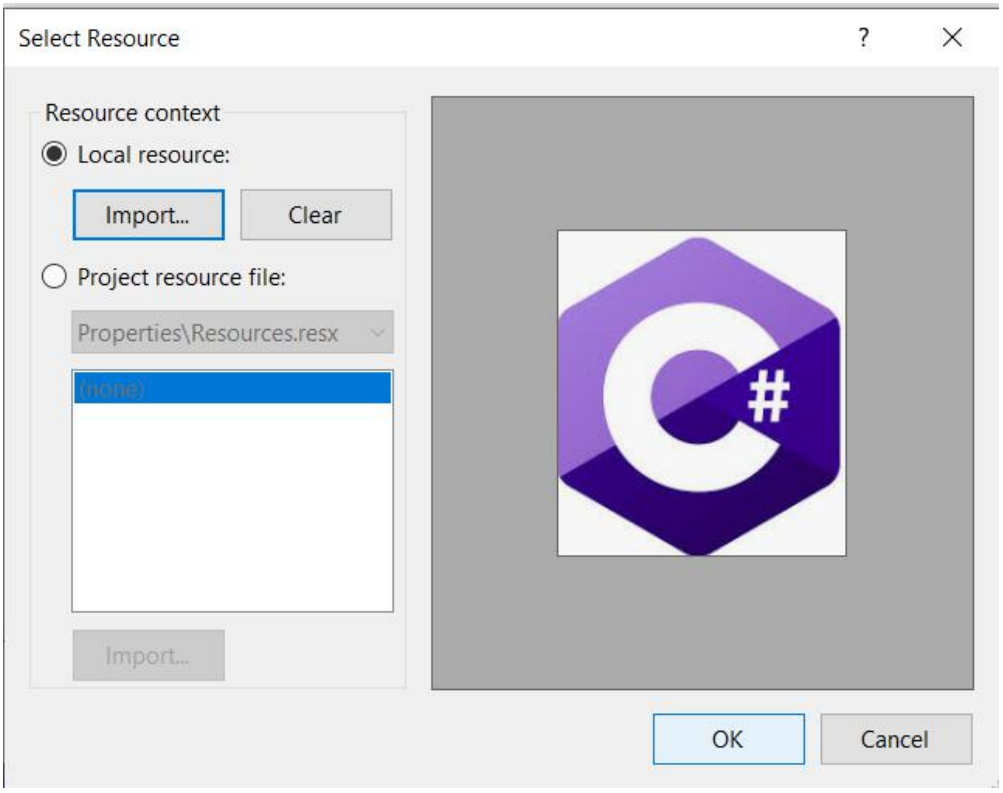


Step 2) The next step is to actually attach an image to the picture box control. This can be done by following the below steps.

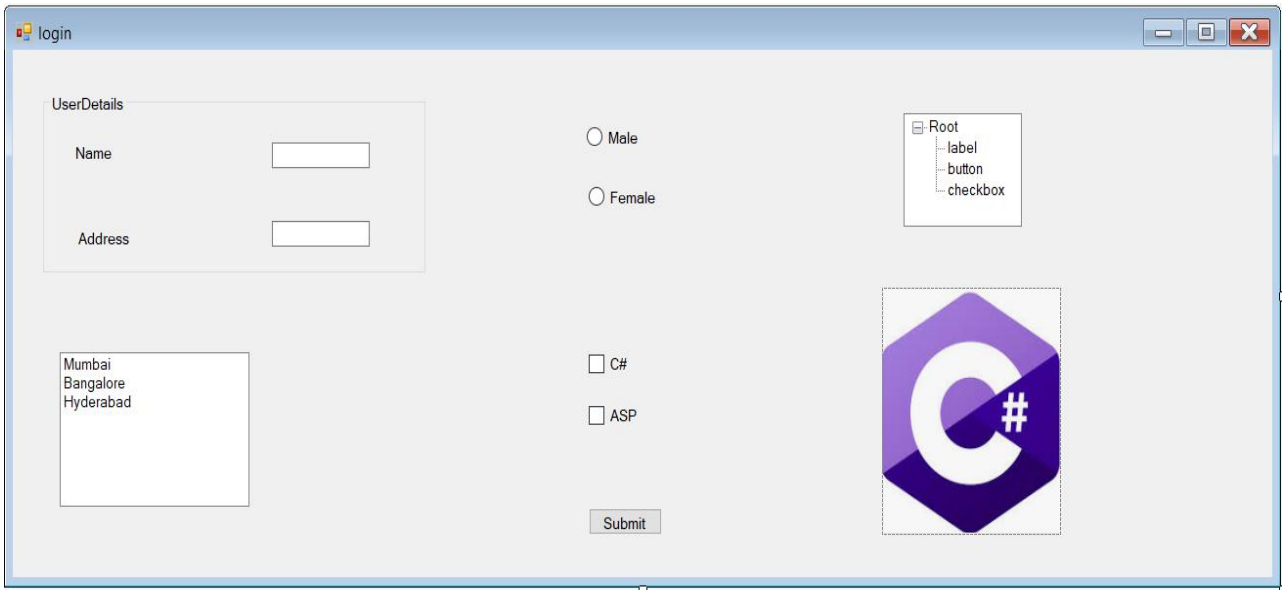
1. First, click on the Image property for the PictureBox control. A new window will pops out.



2. In this window, click on the Import button. This will be used to attach an image to the picturebox control.
3. A dialog box will pop up in which you will be able to choose the image to attach the picturebox 4. Click on the OK button



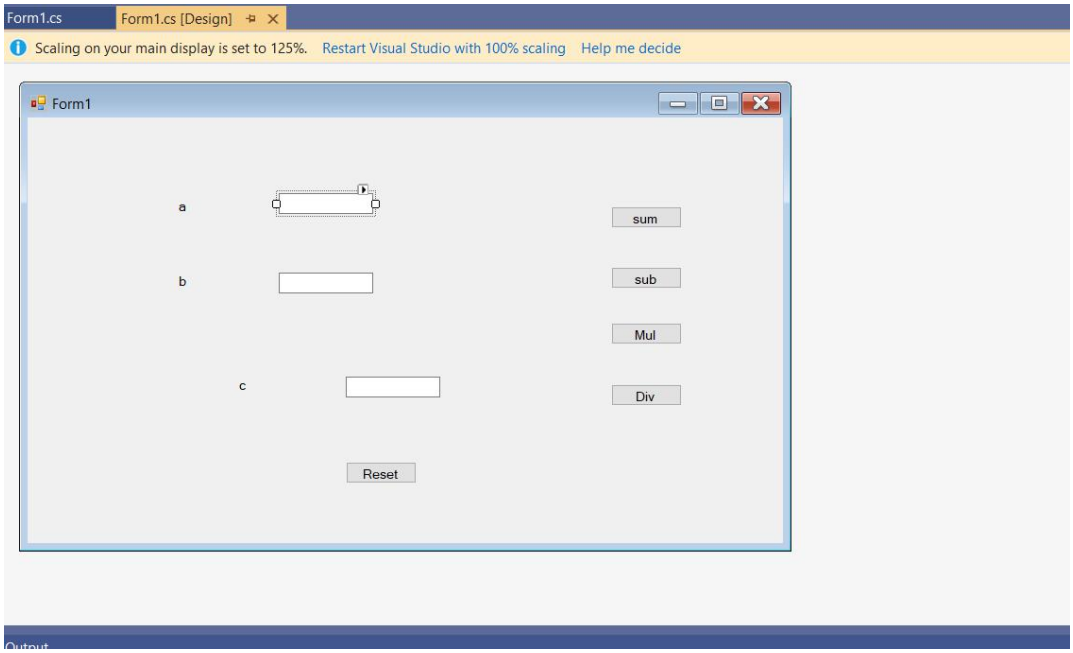
One the changes are made, the output is :



WEEK 7 (21-10-2020)

Calculator Application:

Create Three Text Boxes. Create Three labels and five Buttons , Re-name label1 as a, Re-name label2 as b, Rename label3 as c. Rename buttons as sum , sub , mul , div and reset.

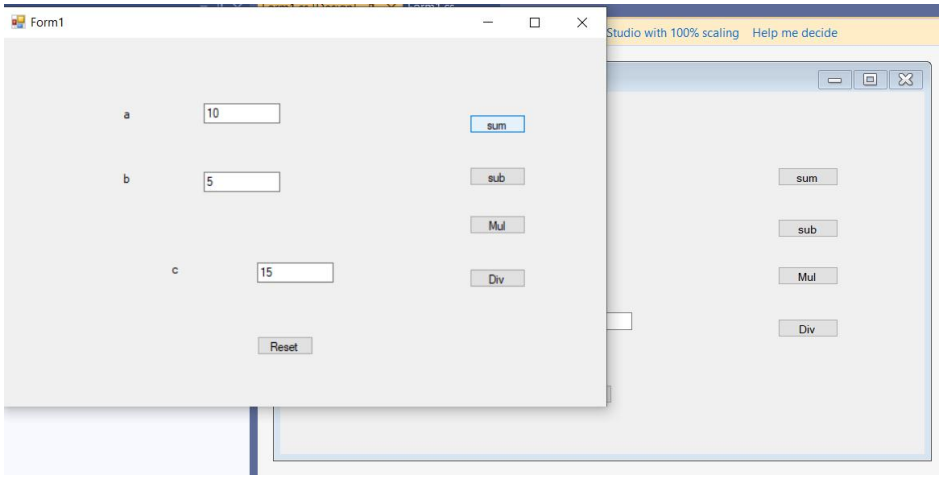


As logic will be in Sum Button ,Double click on Sum Button.

```
int a, b, c;
 a = Convert.ToInt32(textBox1.Text);
b = Convert.ToInt32(textBox2.Text);
c = a + b;
textBox3.Text = c.ToString();
```

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    int a, b, c;
    a = Convert.ToInt32(textBox1.Text);
    b = Convert.ToInt32(textBox2.Text);
    c = a + b;
    textBox3.Text = c.ToString();
}
```

Save and run to get the output.

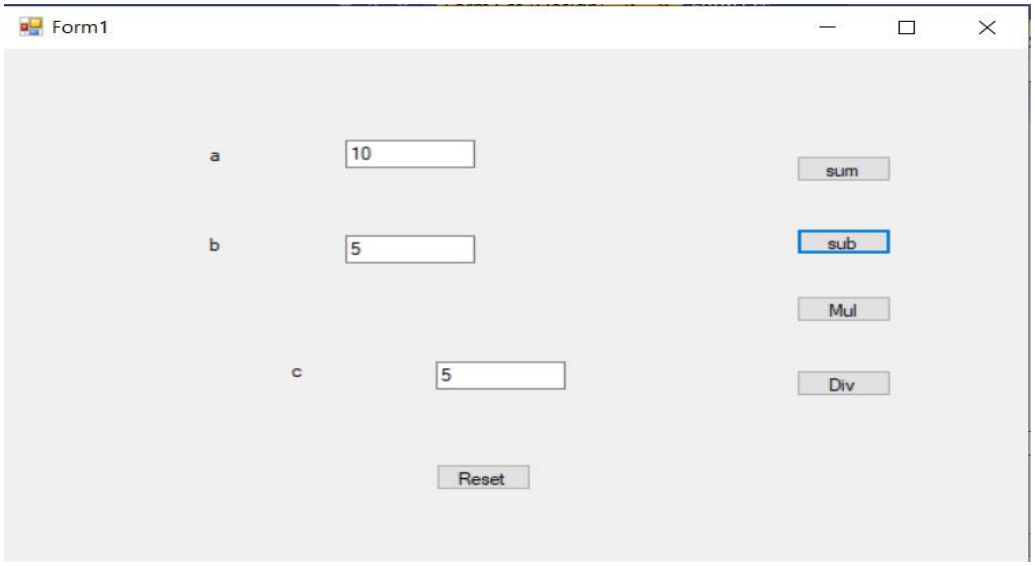


As logic will be in sub Button ,Double click on Sub Button.

```
int a, b, c;
 a = Convert.ToInt32(textBox1.Text);
b = Convert.ToInt32(textBox2.Text);
c = a - b;
textBox3.Text = c.ToString();
```

```
private void button2_Click(object sender, EventArgs e)
{
    int a, b, c;
    a = Convert.ToInt32(textBox1.Text);
    b = Convert.ToInt32(textBox2.Text);
    c = a - b;
    textBox3.Text = c.ToString();
}
```

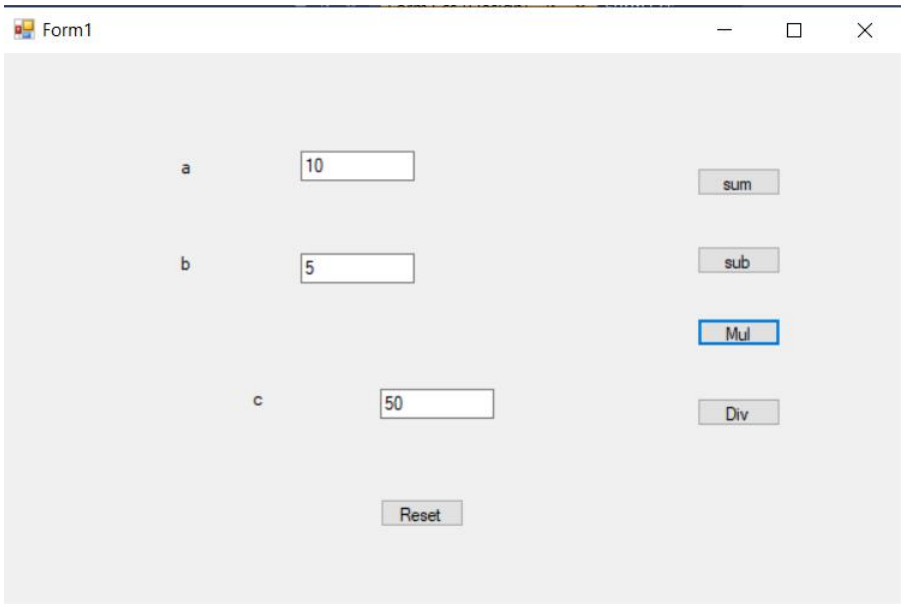
Output:



As logic will be in mul Button ,Double click on mul Button.
int a, b, c;
a = Convert.ToInt32(textBox1.Text);
b = Convert.ToInt32(textBox2.Text);
c = a + b;
textBox3.Text = c.ToString();

```
1 reference
private void button3_Click(object sender, EventArgs e)
{
    int a, b, c;
    a = Convert.ToInt32(textBox1.Text);
    b = Convert.ToInt32(textBox2.Text);
    c = a * b;
    textBox3.Text = c.ToString();
}
```

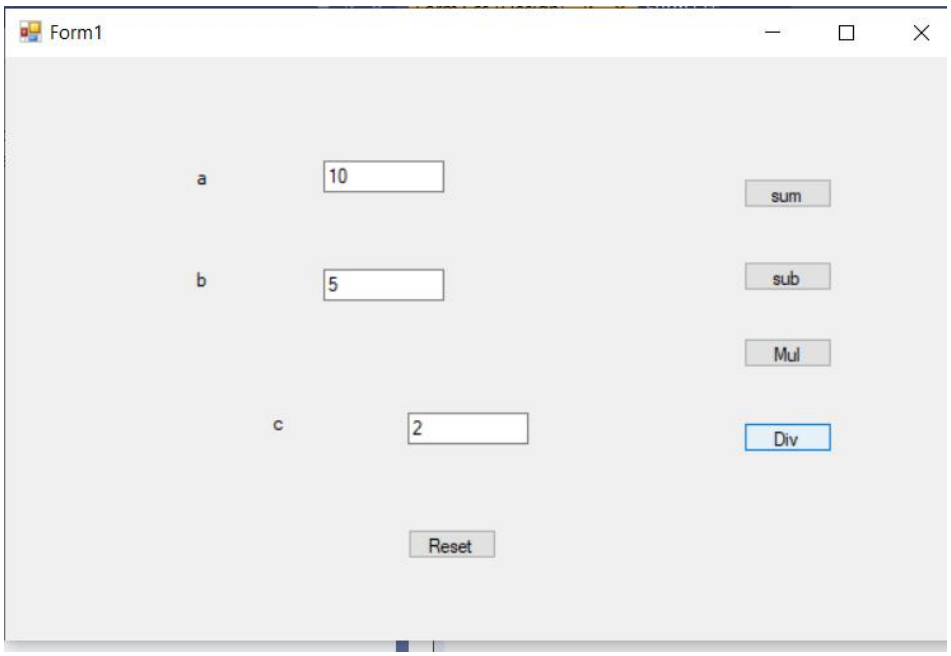
Output:



As logic will be in div Button ,Double click on div Button.
int a, b, c;
a = Convert.ToInt32(textBox1.Text);
b = Convert.ToInt32(textBox2.Text);
c = a + b;
textBox3.Text = c.ToString();

```
1 reference
private void button4_Click(object sender, EventArgs e)
{
    int a, b, c;
    a = Convert.ToInt32(textBox1.Text);
    b = Convert.ToInt32(textBox2.Text);
    c = a / b;
    textBox3.Text = c.ToString();
}
```

Output:

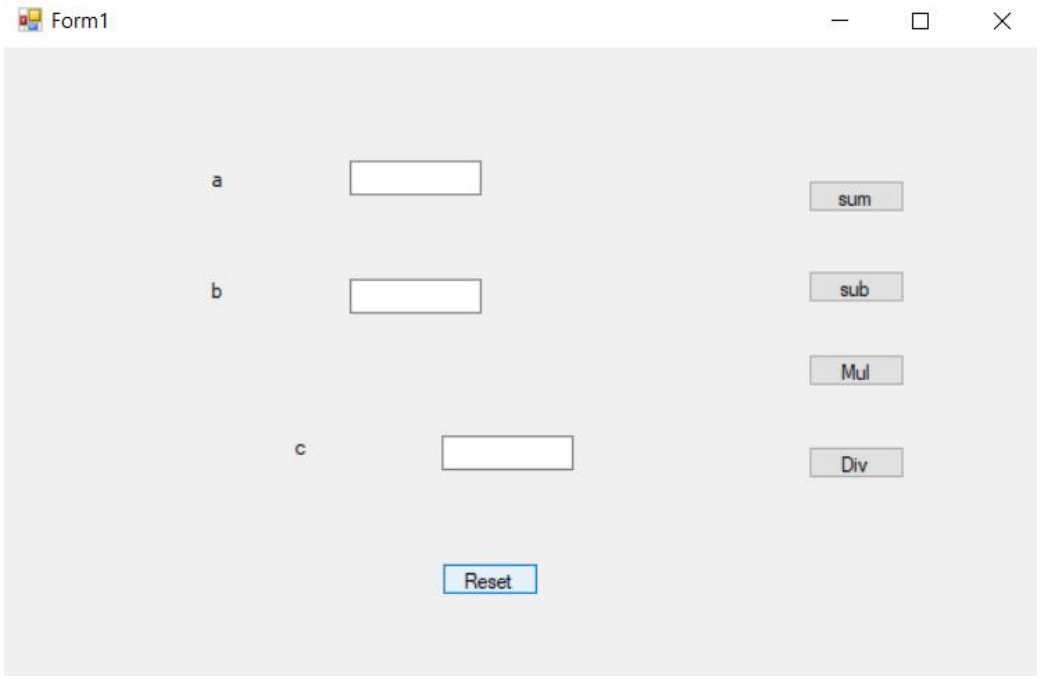


Double click on Clear button.

textBox1.Text = " ";
textBox2.Text = " ";
textBox3.Text = " ";

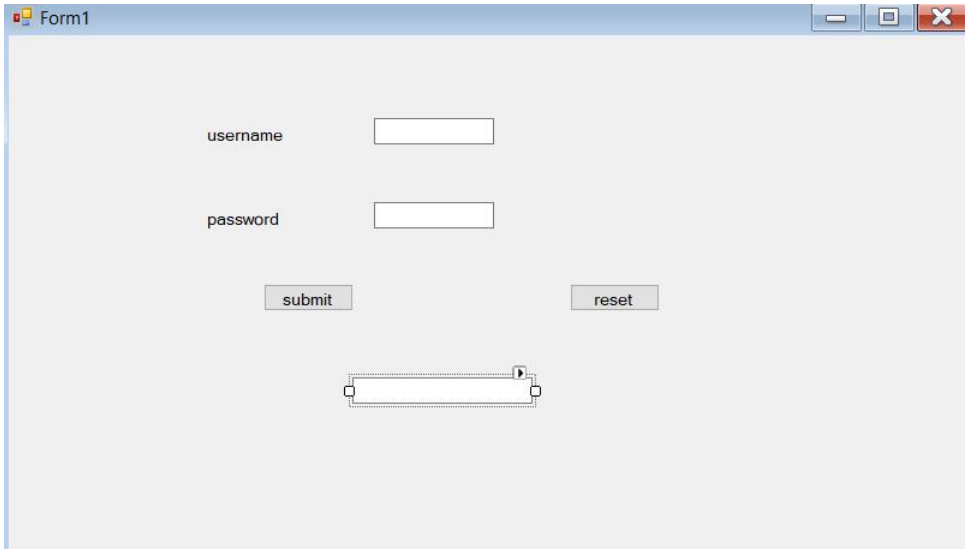
```
1 reference
private void button5_Click(object sender, EventArgs e)
{
    textBox1.Text = " ";
    textBox2.Text = " ";
    textBox3.Text = " ";
}
```

Output:



LOGIN PAGE:

Create Three Text Boxes. Create 2 labels and 2 Buttons , Re-name label1 as username, Re-name label2 as password. Rename buttons as submit and reset.



Right click on the textbox corresponding to password and go to the properties. Enter "*" in passwordchar box .

Modifiers	Private
Multiline	False
PasswordChar	*
ReadOnly	False
RightToLeft	No
ScrollBars	None
ShortcutsEnabled	True
Size	**100, 22**

ext

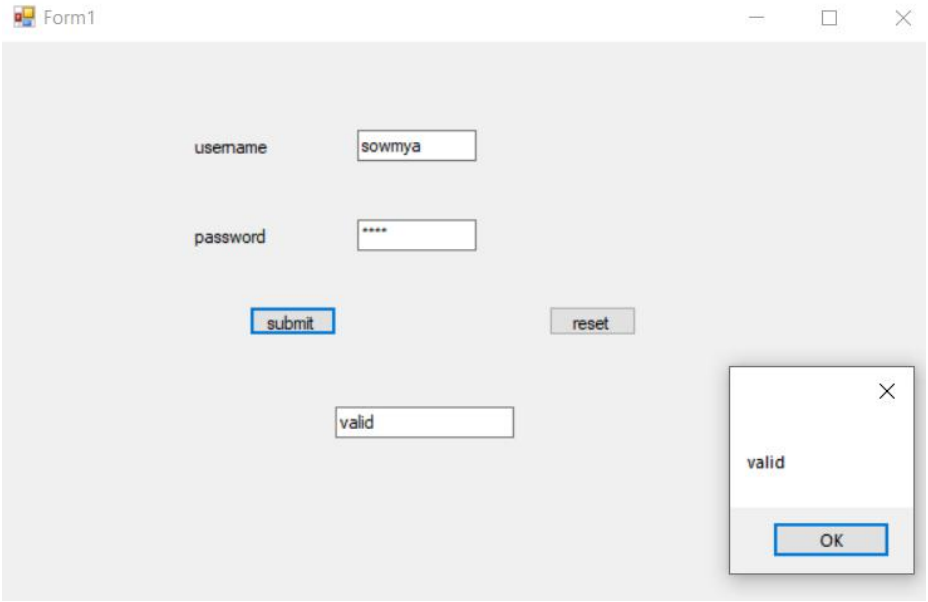
Double click on submit button and enter the code:

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    SetValueForText1 = textBox1.Text;
    SetValueForText2 = textBox2.Text;

    if (SetValueForText1 == "sowmya" && SetValueForText2 == "0504")
    {
        textBox3.Text = "valid";
        MessageBox.Show("valid");
    }
    else
    {
        textBox3.Text = "not valid";
        MessageBox.Show("not valid");
    }
}
```

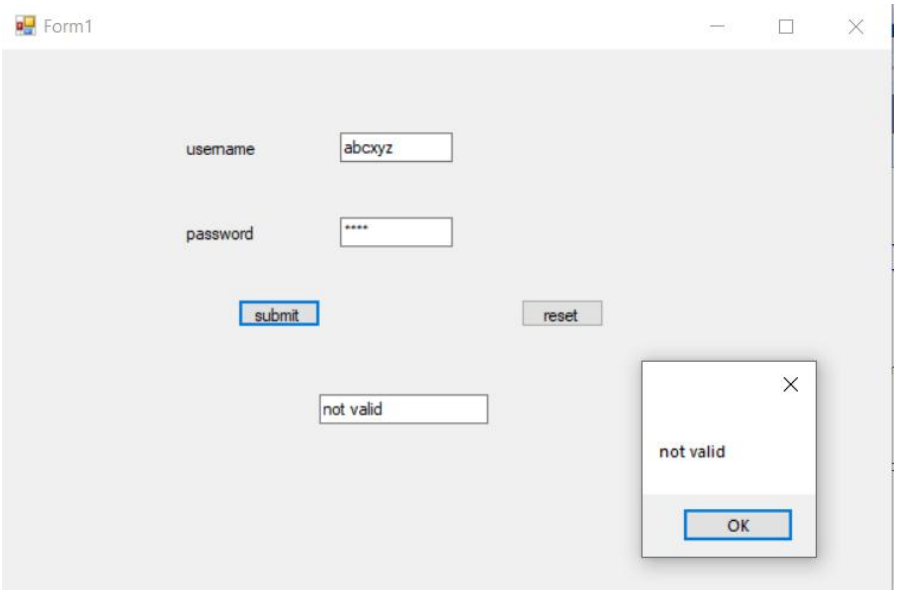
Double click on reset button and enter the code:

```
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = " ";
    textBox2.Text = " ";
}
```

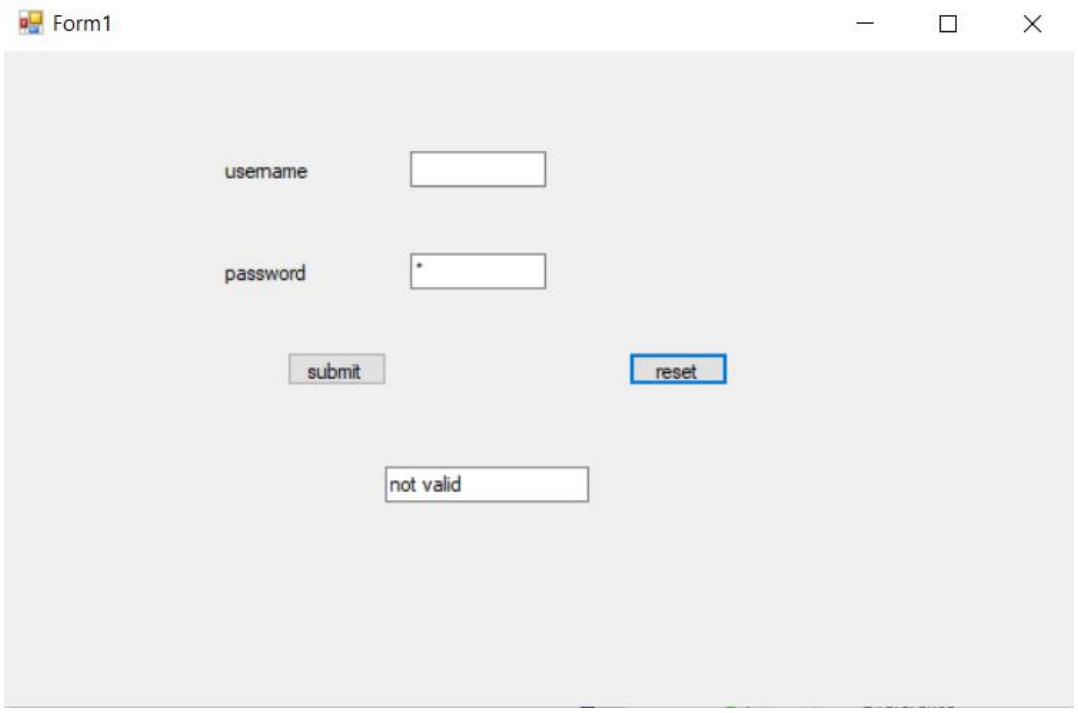
Now,save and run get the output.
When the username and password are entered correctly:



When the username and password are not entered correctly:

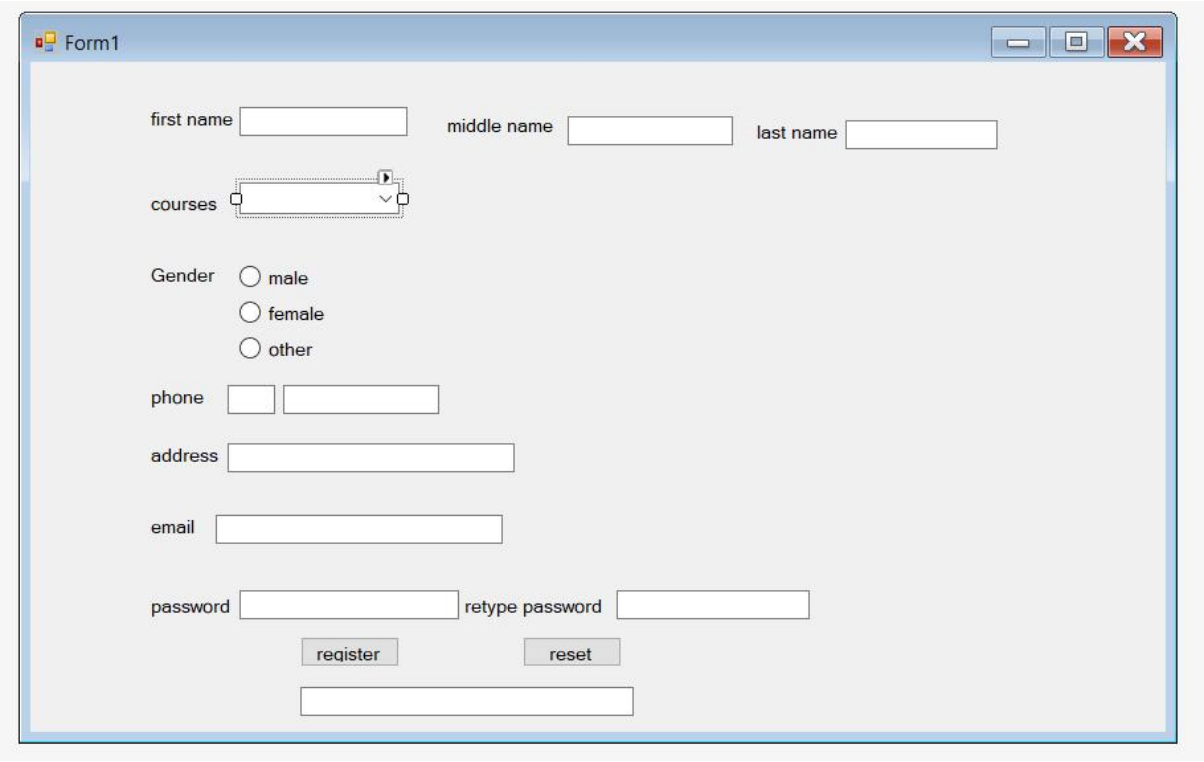


After clicking on reset button:

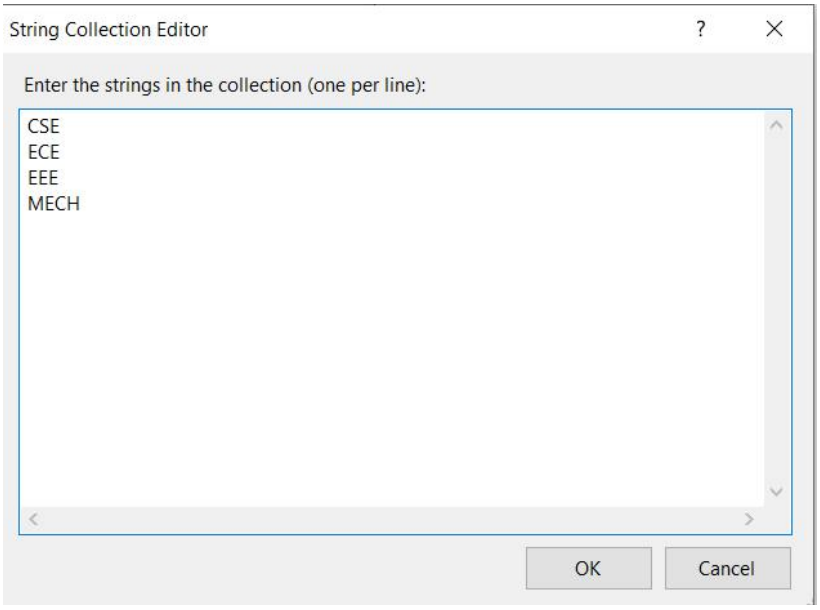


REGISTRATION PAGE:

Drag 10 labels and rename them accordingly , then drag 10 textboxes and 1 combo box and 3 radio buttons and rename radio buttons as male, female and others.Drag 2 buttons and rename them as register and reset.



Go to the properties of combo box and click on items collections. Enter as below and click on ok.



Double click on the register button and enter the code as follows:

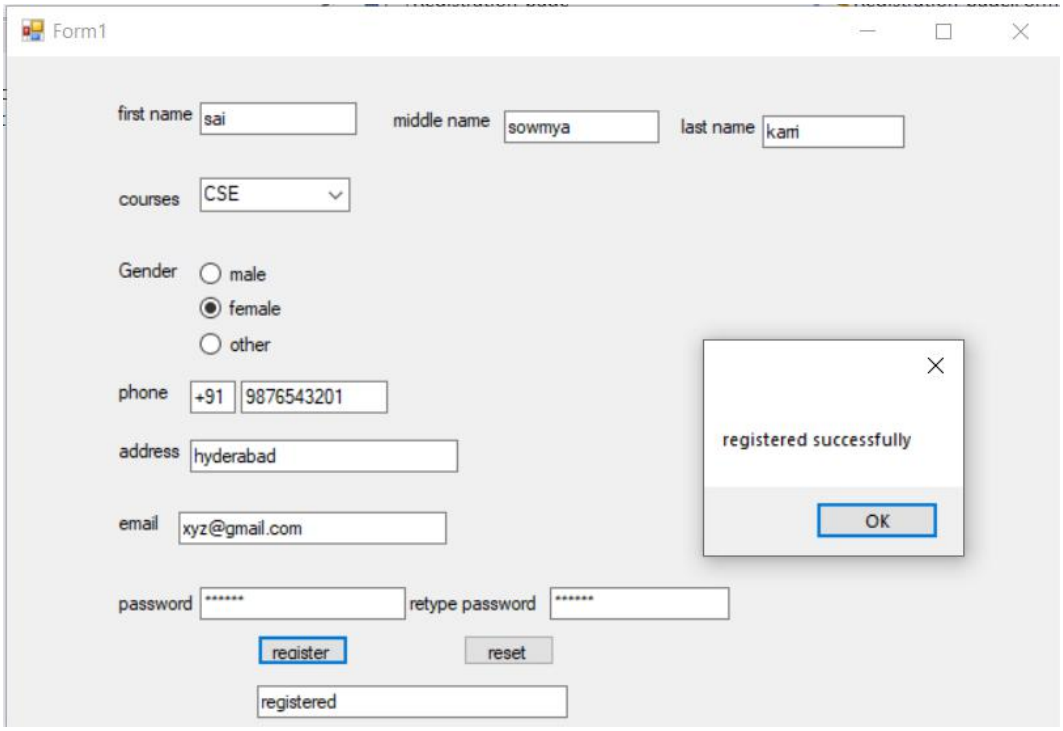
```
private void button1_Click(object sender, EventArgs e)
{
    SetValueForText1 = textBox1.Text;
    SetValueForText2 = textBox2.Text;
    SetValueForText3 = textBox3.Text;
    SetValueForText4 = textBox4.Text;
    SetValueForText5 = textBox5.Text;
    SetValueForText6 = textBox6.Text;
    SetValueForText7 = textBox7.Text;
    SetValueForText8 = textBox8.Text;
    SetValueForText9 = textBox9.Text;
    if (SetValueForText8 == SetValueForText9)
    {
        textBox10.Text = "registered";
        MessageBox.Show("registered successfully");
    }
    else
    {
        textBox10.Text = "password didn't match";
        MessageBox.Show("not registered");
    }
```

Double click on the reset button and enter the code as seen below :

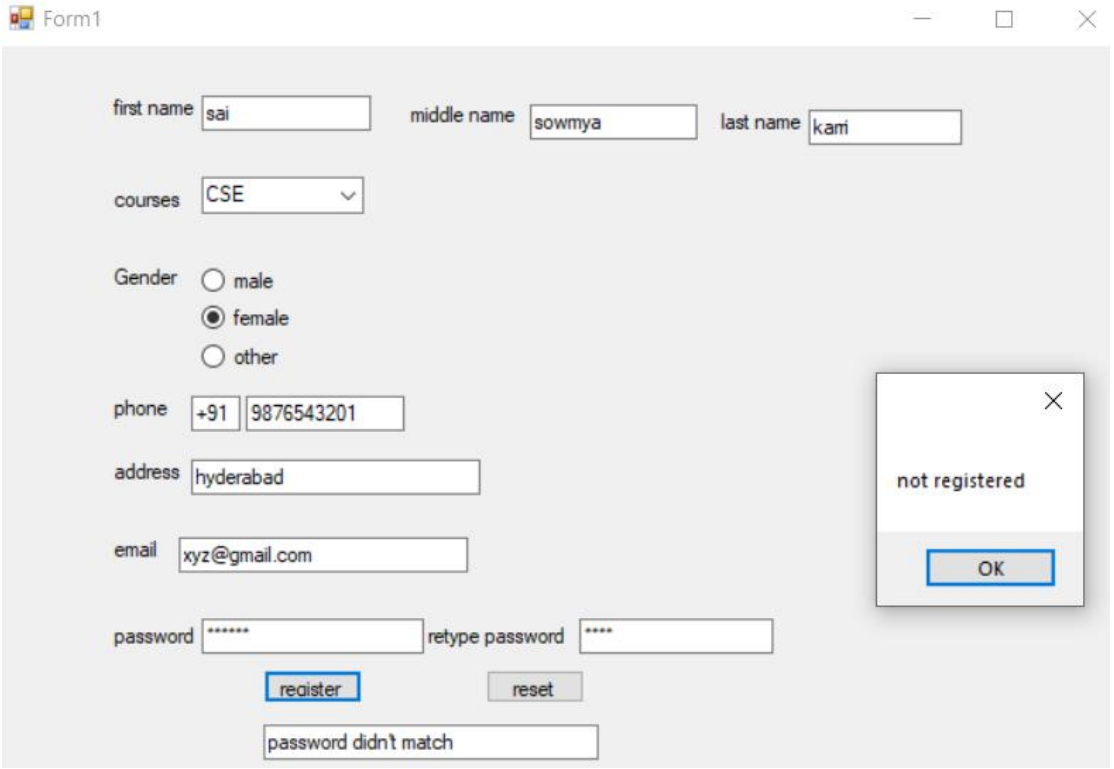
```
    private void button2_Click(object sender, EventArgs e)
    {
        textBox1.Text = " ";
        textBox2.Text = " ";
        textBox3.Text = " ";
        textBox4.Text = " ";
        textBox5.Text = " ";
        textBox6.Text = " ";
        textBox7.Text = " ";
        textBox8.Text = " ";
        textBox9.Text = " ";
        textBox10.Text = " ";
        radioButton1.Checked = false;
        radioButton2.Checked = false;
        radioButton3.Checked = false;
        comboBox1.Text = " ";
    }
}
```

Now save and run to get the output.

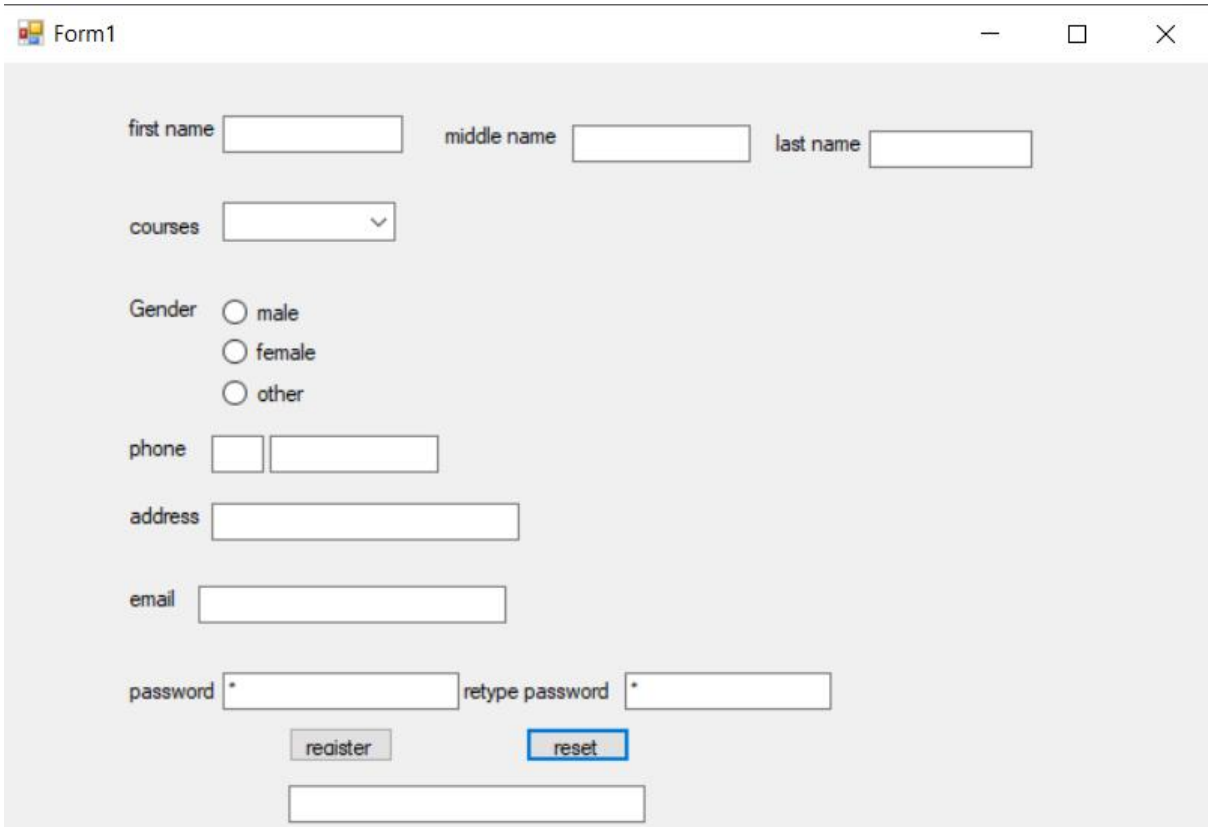
when password and retype password are same :



when password and retype password are not same :



On clicking on reset button:

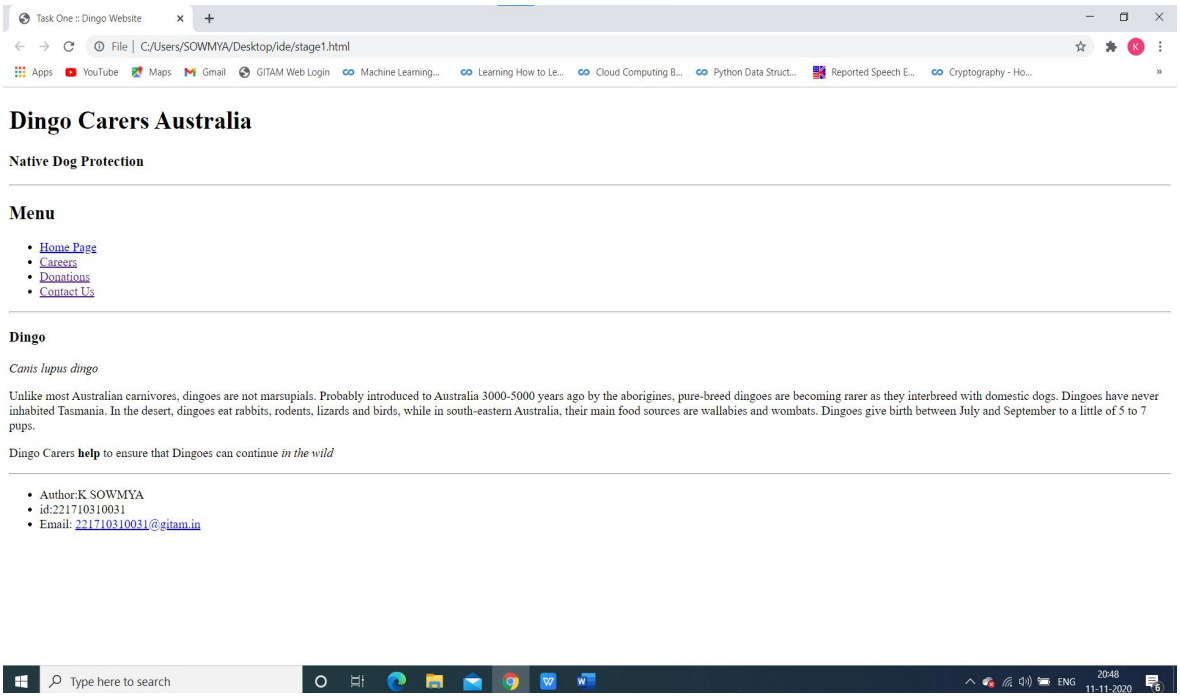


Week-10 (11-11-2020)

HTML and CSS

STAGE 1 :

1. Using well-structured logical ‘XHTML 1.0 Strict’ mark-up, create the below simple webpage.



We create this webpage using Notepad and below code, and save it as a file named ‘stage1.html’.

stage1 - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Task One :: Dingo Website</title>
    </head>
    <body>
        <h1>Dingo Carers Australia</h1>
        <h3>Native Dog Protection</h3>
        <hr>
        <h2>Menu</h2>
        <ul>
            <li><a href="#">Home Page</a></li>
            <li><a href="./underconst.html">Careers</a></li>
            <li><a href="./underconst.html">Donations</a></li>
            <li><a href="./underconst.html">Contact Us</a></li>
        </ul>
        <hr>
        <h3>Dingo</h3>
        <em>Canis lupus dingo</em>
        <p>Unlike most Australian carnivores, dingoes are not marsupials.
          Probably introduced to Australia 3000-5000 years ago by the aborigines,
          pure-breed dingoes are becoming rarer as they interbreed with domestic dogs.
          Dingoes have never inhabited Tasmania. In the desert, dingoes eat rabbits, rodents,
          lizards and birds, while in south-eastern Australia, their main food sources are wallabies and wombats.
          Dingoes give birth between July and September to a little of 5 to 7 pups.</p>
        <p>Dingo Carers <strong>help</strong> to ensure that Dingoes can continue <em>in the wild</em></p>
        <hr>
        <ul>
            <li>Author:K SOWMYA</li>
            <li>id:221710310031</li>
            <li>Email:  <a href="#">221710310031@gitam.in</a></li>
        </ul>
    </body>
</html>
```

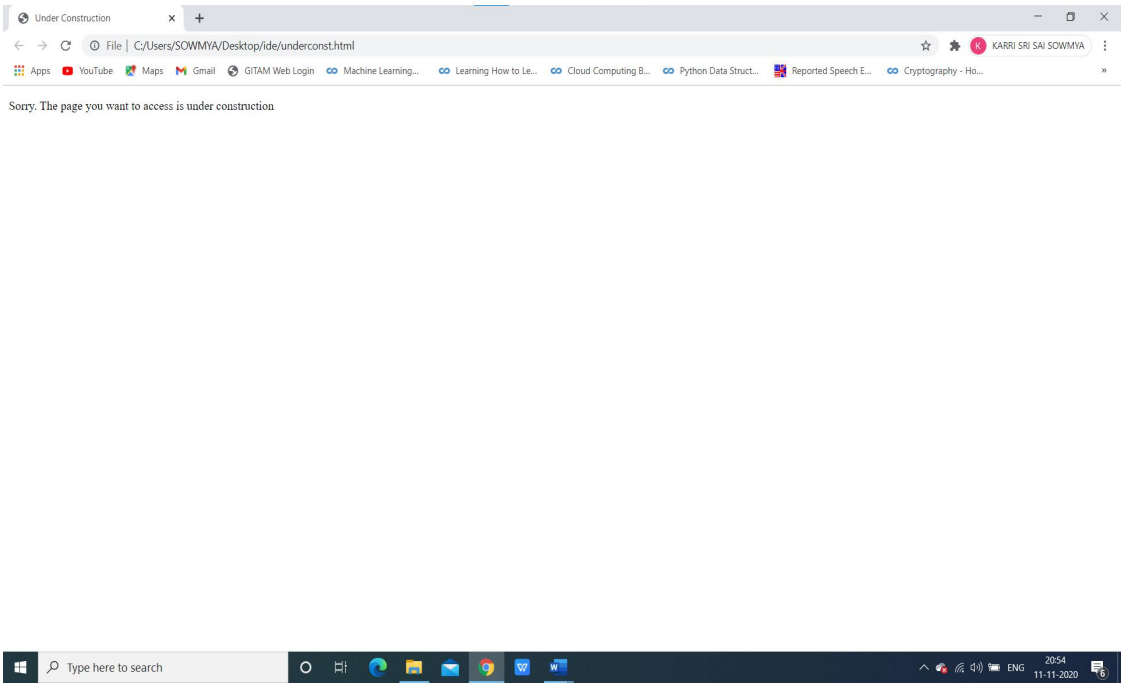
2. Make all the links to another page called ‘underconst.html’ that contains the content: ‘Sorry. The page you are wanting to access is under construction’.

underconst - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Under Construction</title>
    </head>
    <body>
        <p>Sorry. The page you want to access is under construction</p>
    </body>
</html>
```

Webpage output :



3. View your web page in Mozilla Firefox or Internet Explorer, and check that the links work.

STAGE 2 :

1. Save the page as ‘stage2.htm’ and then add the following style, using an embedded style sheet: (that is, add style rules between style tags placed in the head)
   Apply the CSS font-family property, with value of sans-serif, to the body element.
   Give the webpage a background color of #ffffcc.
   Using the CSS color property, change the main heading color to ‘navy’, the secondary headings to ‘blue’, and all emphasized text to ‘red’.
   Align the text in the last paragraph to the right.
2. Add some HTML comments near the beginning of the file to indicate the author , and to indicate when we last validated the page (today).
3. Check that ‘underconst.htm’ file is also well coded and valid.
4. Validate the webpage again.
5. At the beginning of the first main paragraph, add an image element linking to ‘dingo.jpg’ that shows a ‘Photograph of Dingo (Canis lupus dingo)’. The image is 250 wide pixels and 250 pixels high. Add a ‘tooltip’ of ‘Dingo. Photo: Healesville Sanctuary’.
6. Apply style to the image element using ‘float: right’ and ‘border: black solid 2px’
7. Validate the webpage again.

Code :

*stage2 - Notepad

File Edit Format View Help

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
    <head>
        <meta charset="utf-8">
        <title>Task One :: Dingo Website</title>
        <style>
            body{
                font-family: sans-serif;
                background-color: #ffffcc;
            }
            h1{
                color: navy;
            }
            h2{
                color: blue;
            }
            em{
                color: red;
            }

            .tooltip span{
                display:none;
                position:absolute;
            }
            .tooltip:hover span{
                display: block;
            }
            .dingo-image{
                float: right;
                border: black solid 2px;
            }
        </style>
    </head>
```

```
    <body>
        <h1>Dingo Carers Australia</h1>
        <h3>Native Dog Protection</h3>
        <hr>
        <h2>Menu</h2>
        <ul>
            <li><a href="#">Home Page</a></li>
            <li><a href="./underconst.html">Careers</a></li>
            <li><a href="./underconst.html">Donations</a></li>
            <li><a href="./underconst.html">Contact Us</a></li>
        </ul>
        <hr>
        <h3>Dingo</h3>
        <em>Canis lupus dingo</em>
        <div class="tooltip">
            <img class="dingo-image" src="./dingo.jpg" width="250" height="250">
            <span>Dingo. Photo: Healesville Sanctuary</span>
        </div>

        <p>Unlike most Australian carnivores, dinoes are not marsupials. Probably introduced to Australia
        3000-5000 years ago by the aborigines, pure-bred dingoes are becoming rarer as they interbreed with
    domestic dogs.</p>
        <p>Dingo Carers <strong>help</strong> to ensure that Dingoes can continue <em>in the wild</em></p>
        <hr>
        <ul>
           <li>Author:K SOWMYA</li>
            <li>id:221710310031</li>
            <li>Email:  <a href="#">221710310031@gitam.in</a></li>
        </ul>
    </body>
</html>
```

Output webpage :