***COMSATS UNIVERSITY ISLAMABAD ATTOCK CAMPUS***

***COMPUTER SCIENCE (SOFTWARE ENGINEERING)***

***LAB ASSIGNMENT NO : 01***

***SUBJECT:*** ***VISUAL PROGRAMMING***

***SUBMITTED BY:*** ***RAVIA IQBAL***

***REGISTRATION NO:*** ***SP21-BSE-025/ATK***

***SUBMITTED TO:*** ***SIR JAMAL***

***SEMESTER:*** ***05***

***DATE:*** ***19TH MARCH, 2023***

**Problem 01:**

In the following program for printing numbers from 0 to 9, the loop must break when the value of the number becomes 5. Place the break statement at the appropriate position to achieve the result.

using System; {

public class StopCounter {

public static void Main() {

for (int index=0; index < 10; index++) {

Console.WriteLine("Counter Index: (0)", index);

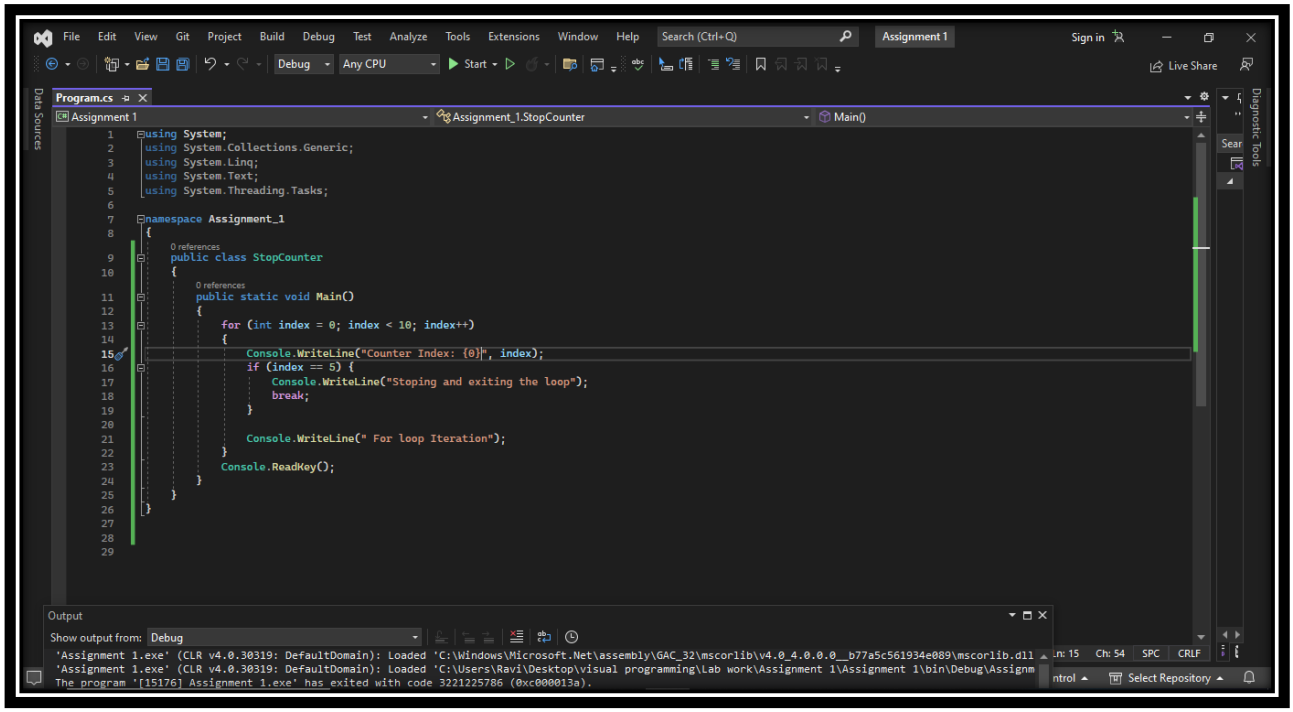
if (index==5) {{ }}

Console.WriteLine("Stoping and exiting the loop");

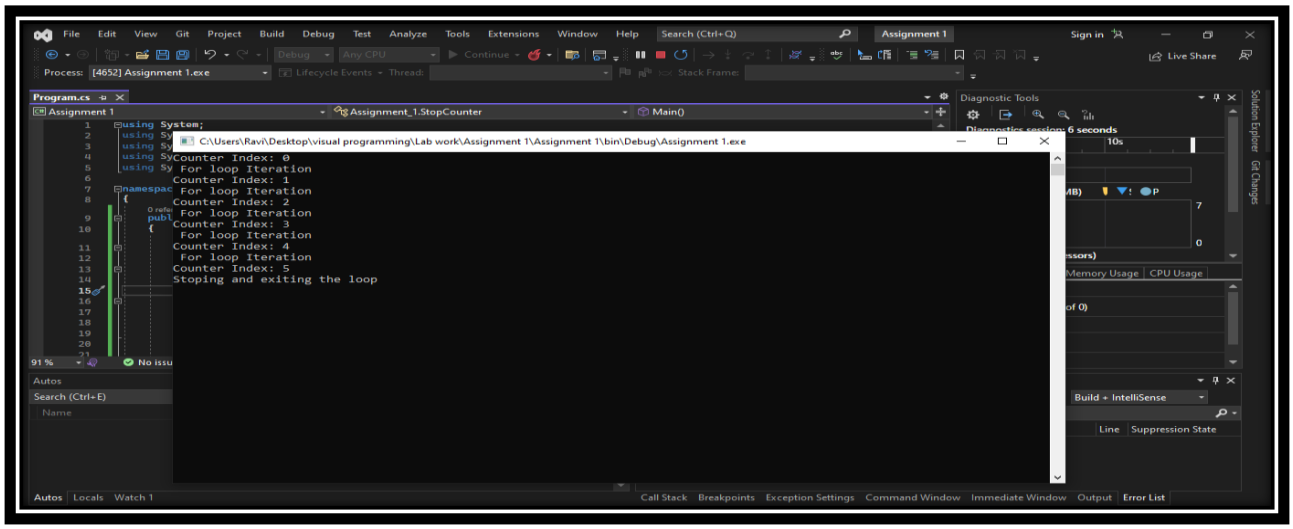
Console.WriteLine(" For loop Iteration");

}}}

**Solution:**



## Output:

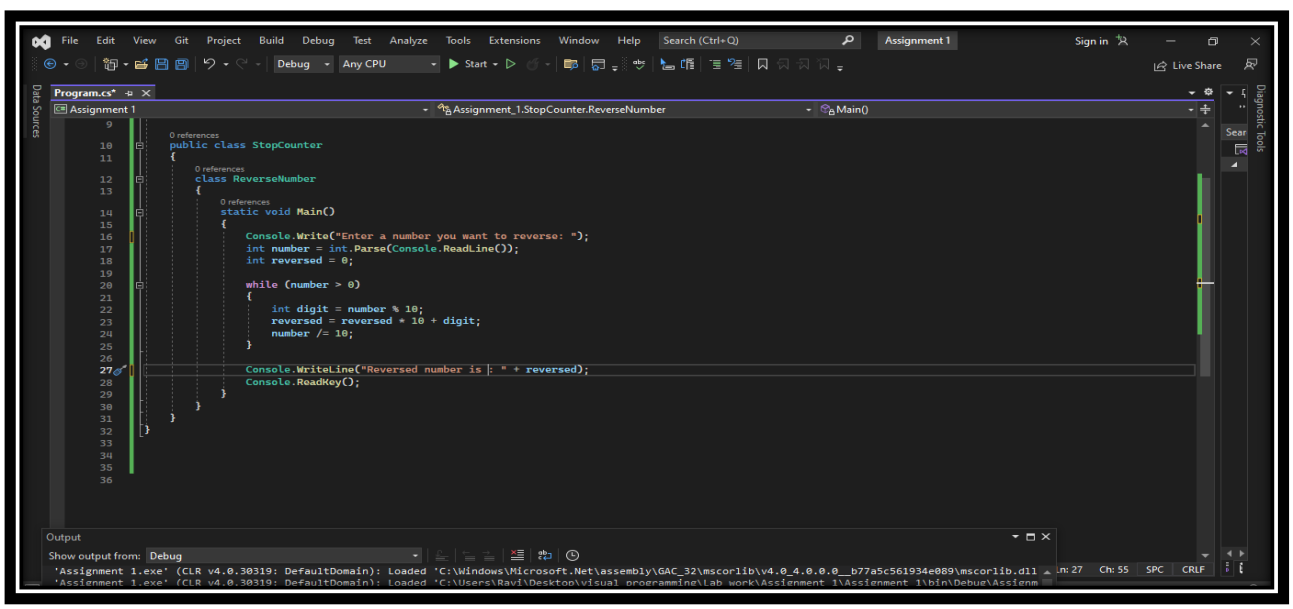



In this program, the break statement is added inside the if statement that checks if the value of index is equal to 5. When index is equal to 5, the break statement is executed, which immediately exits the loop.Also, in the original program, there are two pairs of curly braces around the if statement, which are unnecessary and cause a syntax error. They have been removed in the corrected program.

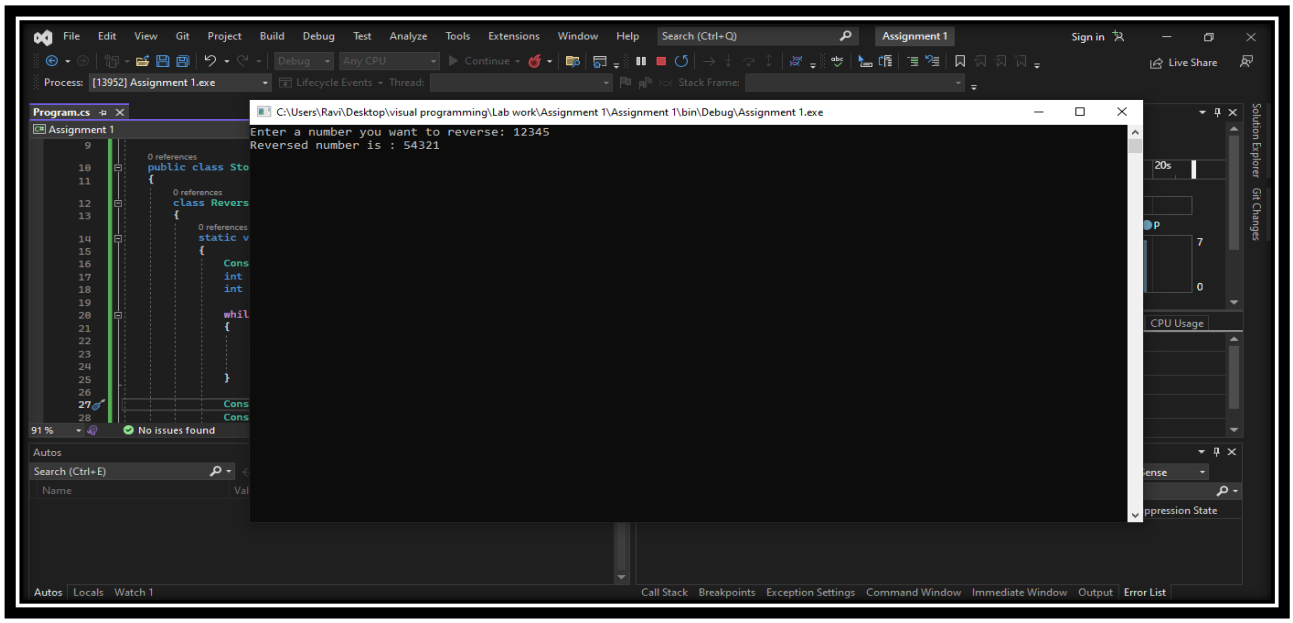
## Program 2:

Given a number, write a program using while loop to reverse the digits of the number. For example, the number 12345 should be written as 54321.

## Solution:



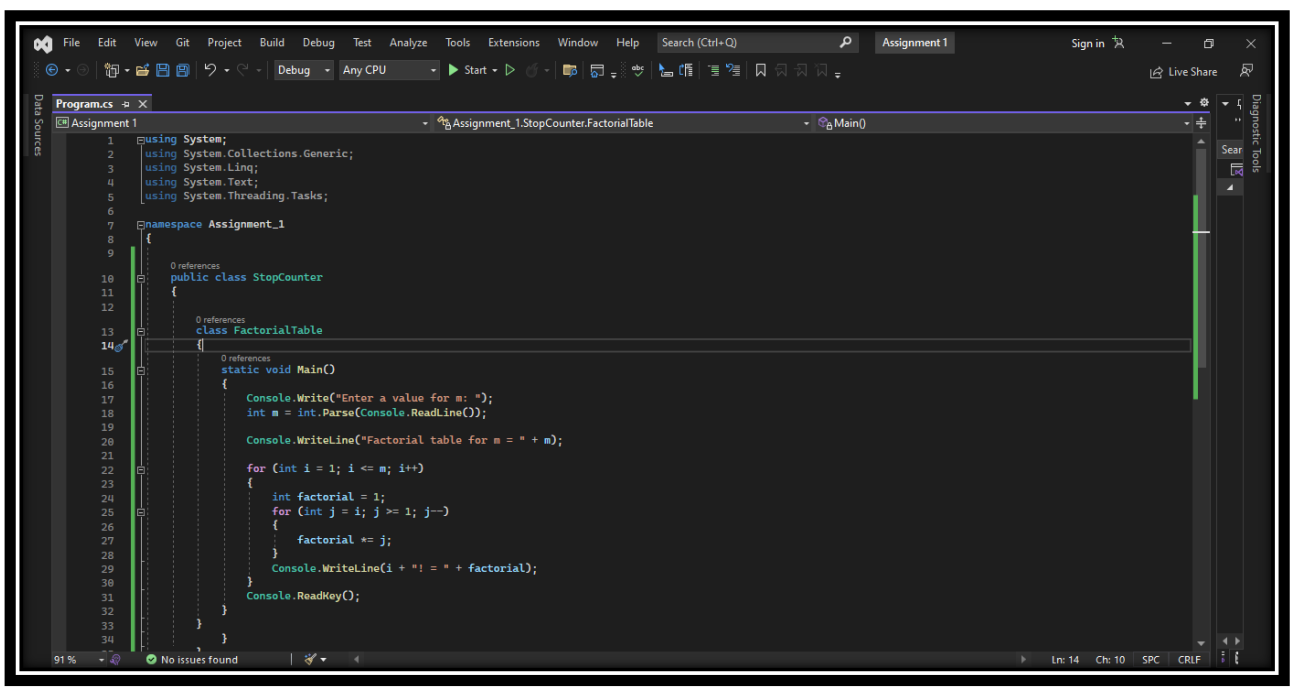
## Output:



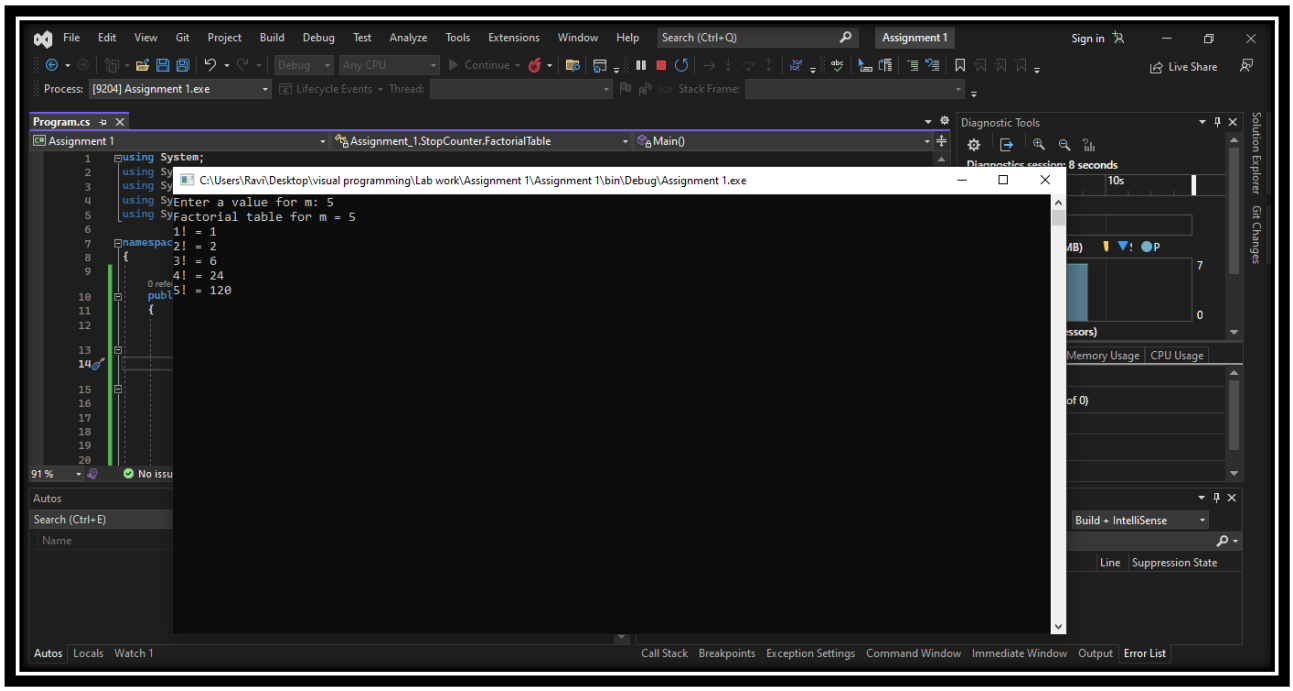

## Problem 3:

The factorial of an integer m is the product of consecutive integers from 1 to m. That is Factorial m = ml=m*(m-1)"..."1. Write a program that computes and prints a table of factorials for any given m.

Solution:

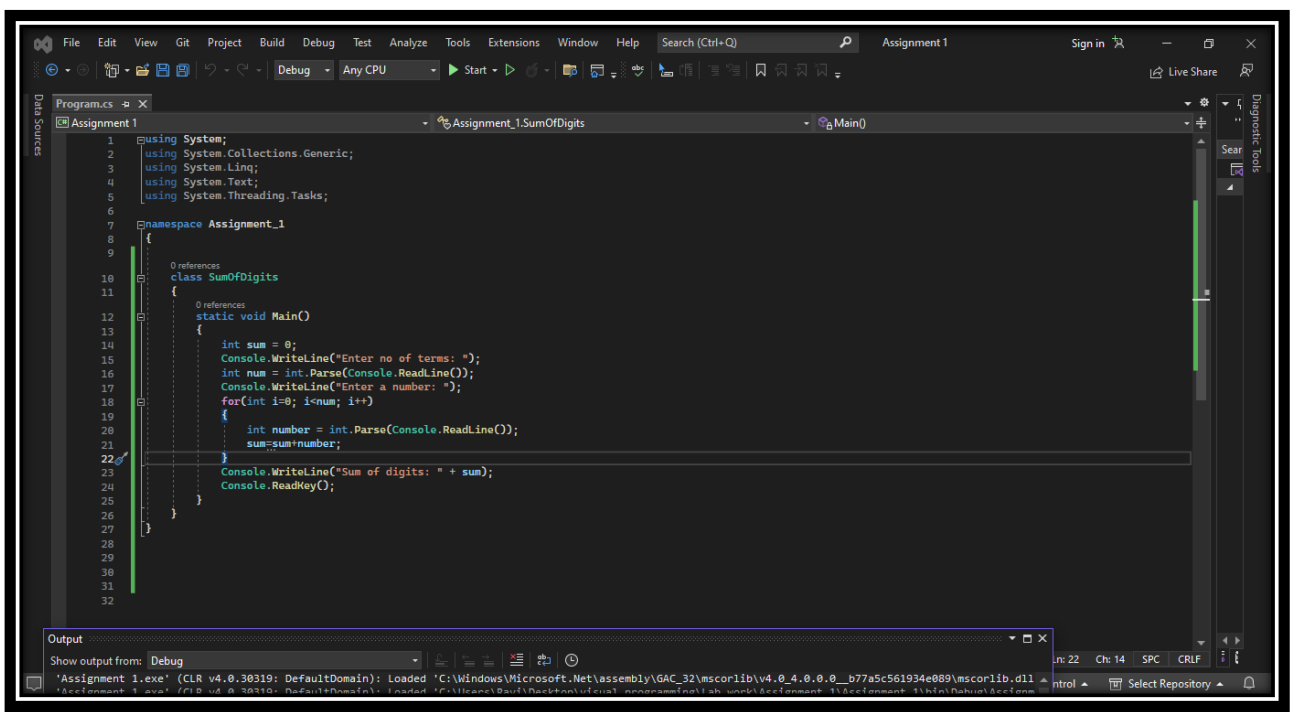


## Output:

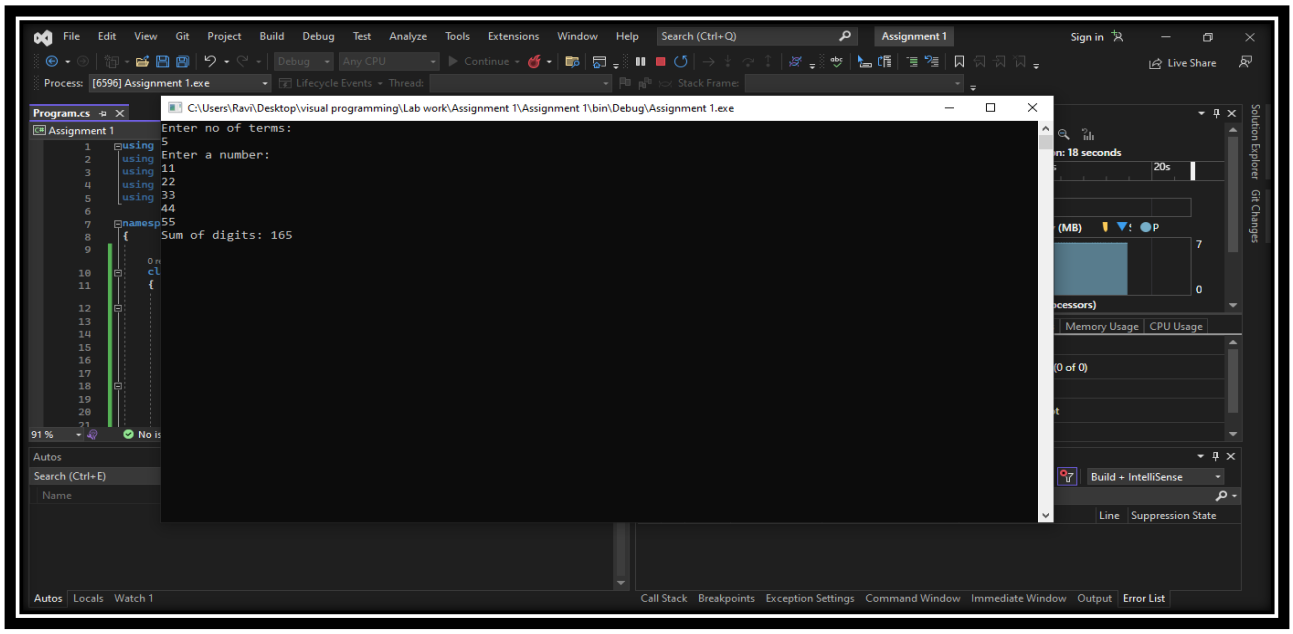



## Program 4:

Write a program to compute the sum of the digits of a given integer number.



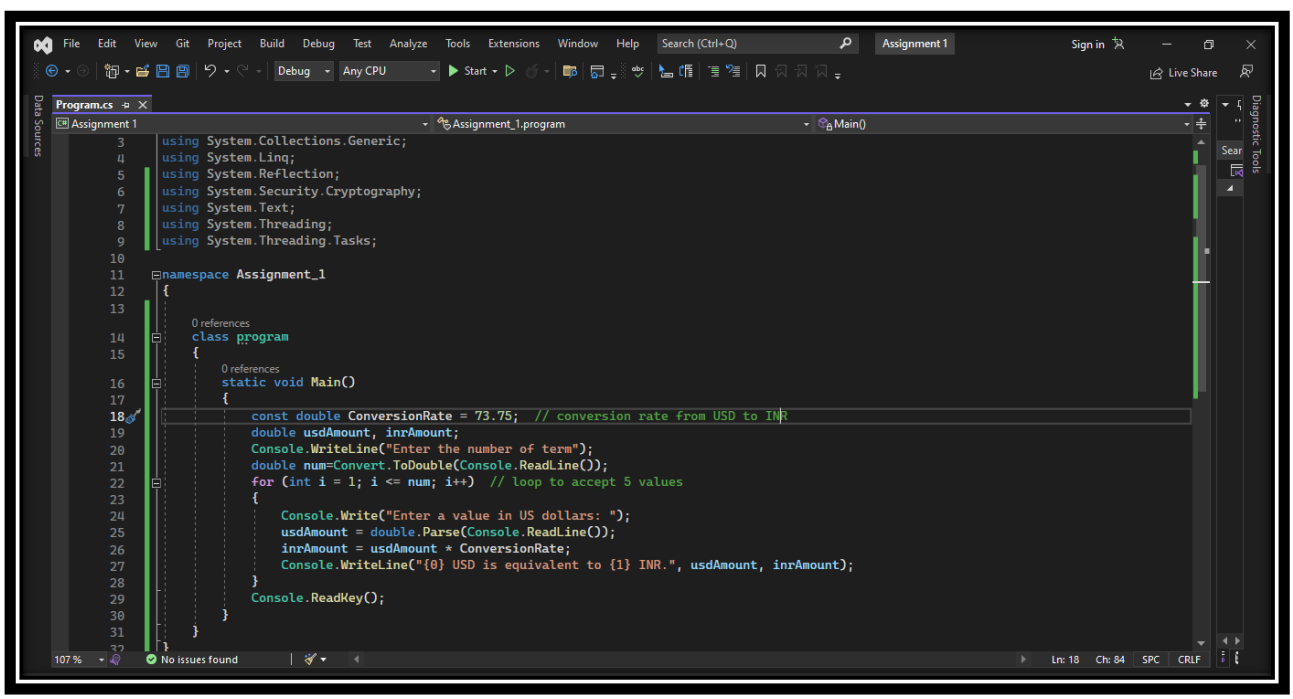
## Output:




## Program 5:

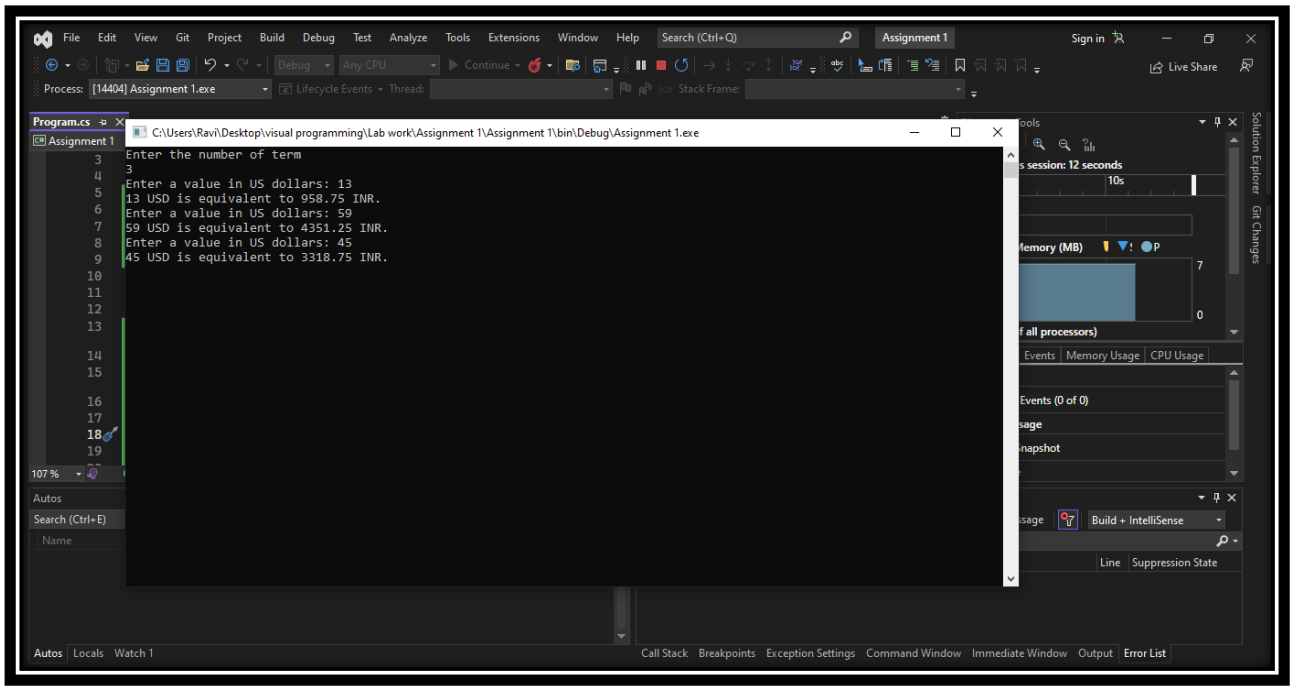
Write a program using a for that accepts five values in US dollars, one at a time, and converts each value entered to its Indian rupees equivalent before the next value is requested.

## Solution:



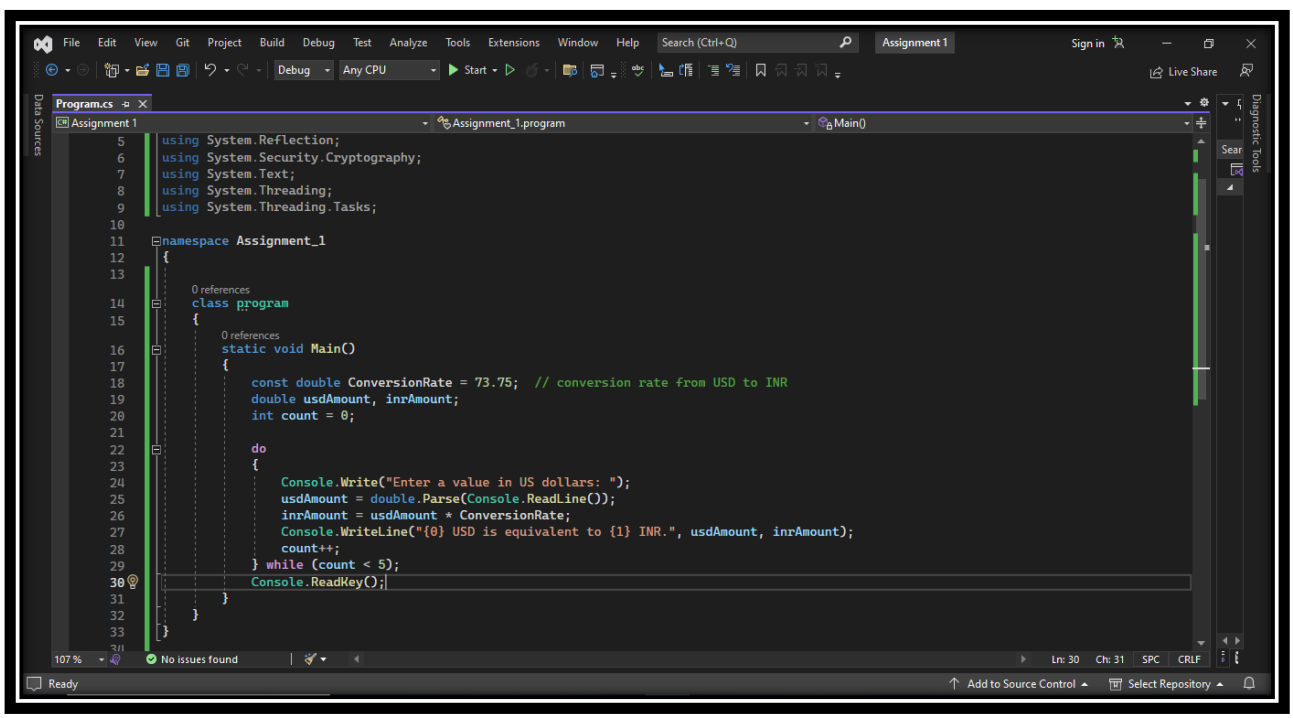
```
using System.Collections.Generic;
using System.Linq;
using System.Reflection;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Assignment_1
{

    class program
    {
        static void Main()
        {
            const double ConversionRate = 73.75;  // conversion rate from USD to INR
            double usdAmount, inrAmount;
            Console.WriteLine("Enter the number of term");
            double num=Convert.ToDouble(Console.ReadLine());
            for (int i = 1; i <= num; i++)  // loop to accept 5 values
            {
                Console.Write("Enter a value in US dollars: ");
                usdAmount = double.Parse(Console.ReadLine());
                inrAmount = usdAmount * ConversionRate;
                Console.WriteLine("{0} USD is equivalent to {1} INR.", usdAmount, inrAmount);
            }
            Console.ReadKey();
        }
    }
}
```

## Output:



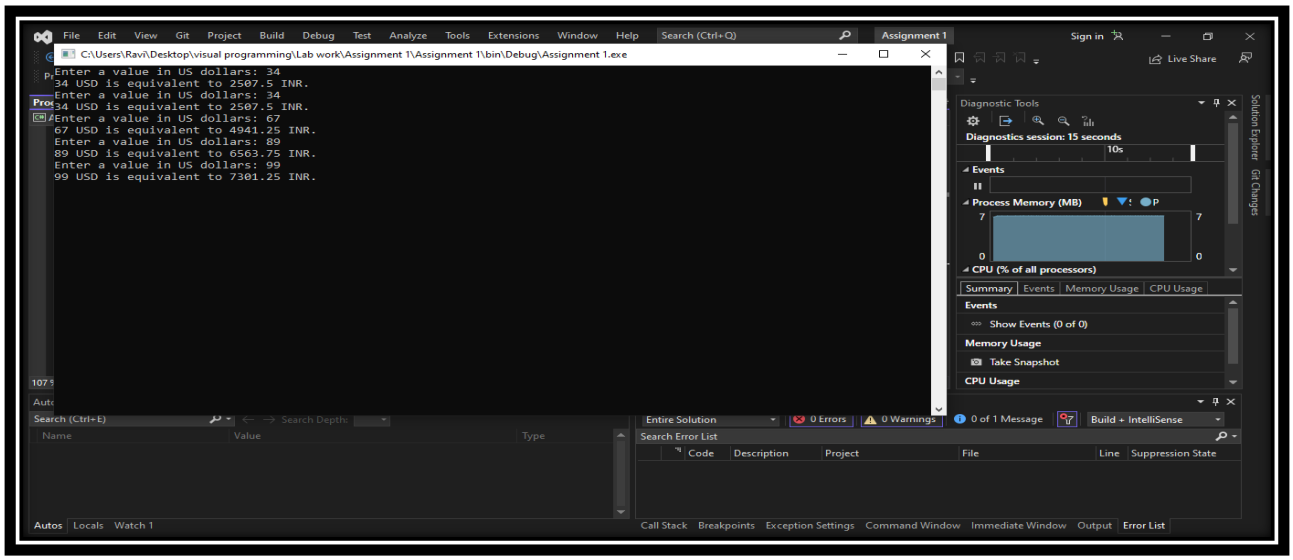

## Program 6:

Repeat the Problem 9 using a do statement.



```
using System.Reflection;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Assignment_1
{

    0 references
    class program
    {
        0 references
        static void Main()
        {
            const double ConversionRate = 73.75;  // conversion rate from USD to INR
            double usdAmount, inrAmount;
            int count = 0;

            do
            {
                Console.Write("Enter a value in US dollars: ");
                usdAmount = double.Parse(Console.ReadLine());
                inrAmount = usdAmount * ConversionRate;
                Console.WriteLine("{0} USD is equivalent to {1} INR.", usdAmount, inrAmount);
                count++;
            } while (count < 5);
            Console.ReadKey();
        }
    }
}
```

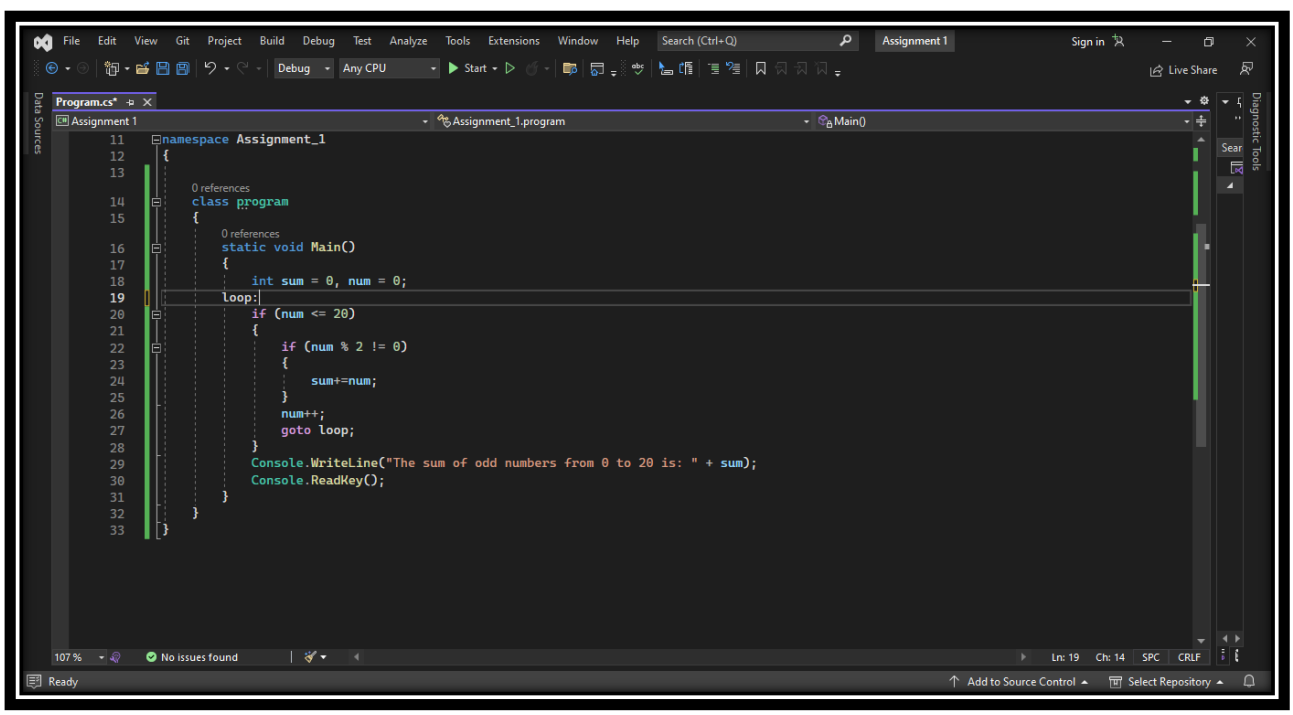
## Output:



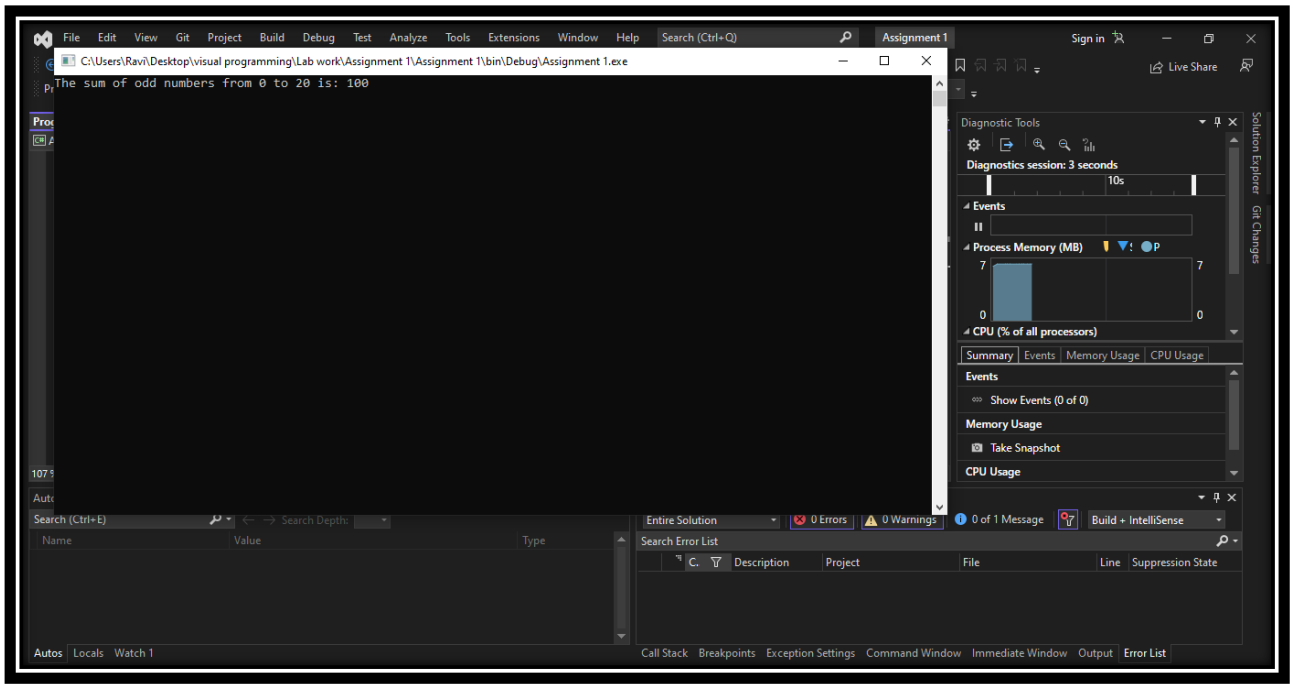

## Program 7:

Write a program to add all the odd numbers from 0 to 20. Use a simple if and goto statements to form a loop of operations

## Solution:

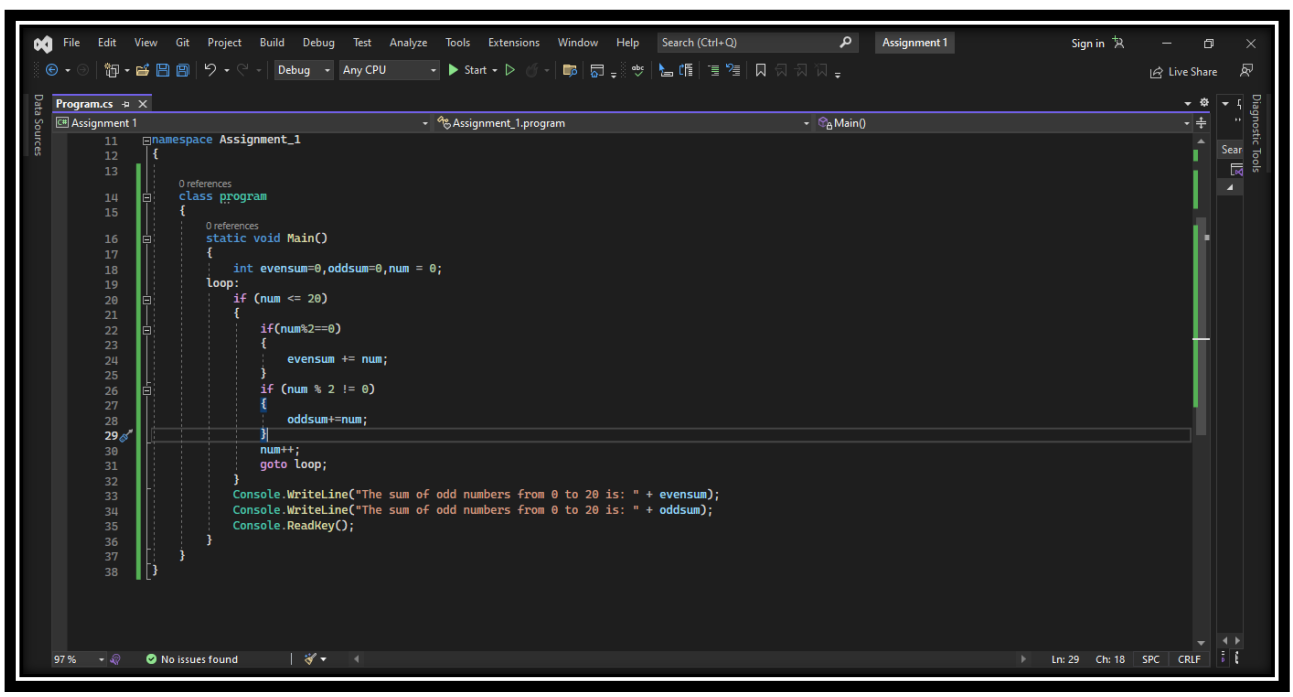


## Output:



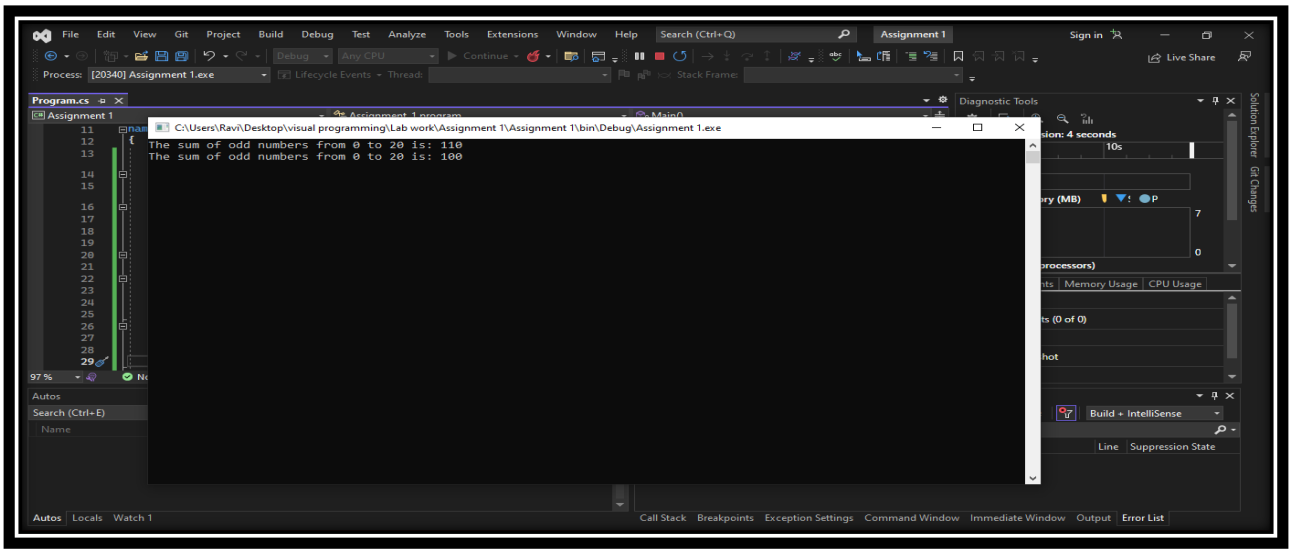

## Problem 8:

Modify the above program so that it also adds all the even numbers between 0 and 20



```
namespace Assignment_1
{

    class program
    {
        static void Main()
        {
            int evensum=0,oddsum=0,num = 0;
        loop:
            if (num <= 20)
            {
                if(num%2==0)
                {
                    evensum += num;
                }
                if (num % 2 != 0)
                {
                    oddsum+=num;
                }
                num++;
                goto loop;
            }
            Console.WriteLine("The sum of odd numbers from 0 to 20 is: " + evensum);
            Console.WriteLine("The sum of odd numbers from 0 to 20 is: " + oddsum);
            Console.ReadKey();
        }
    }
}
```

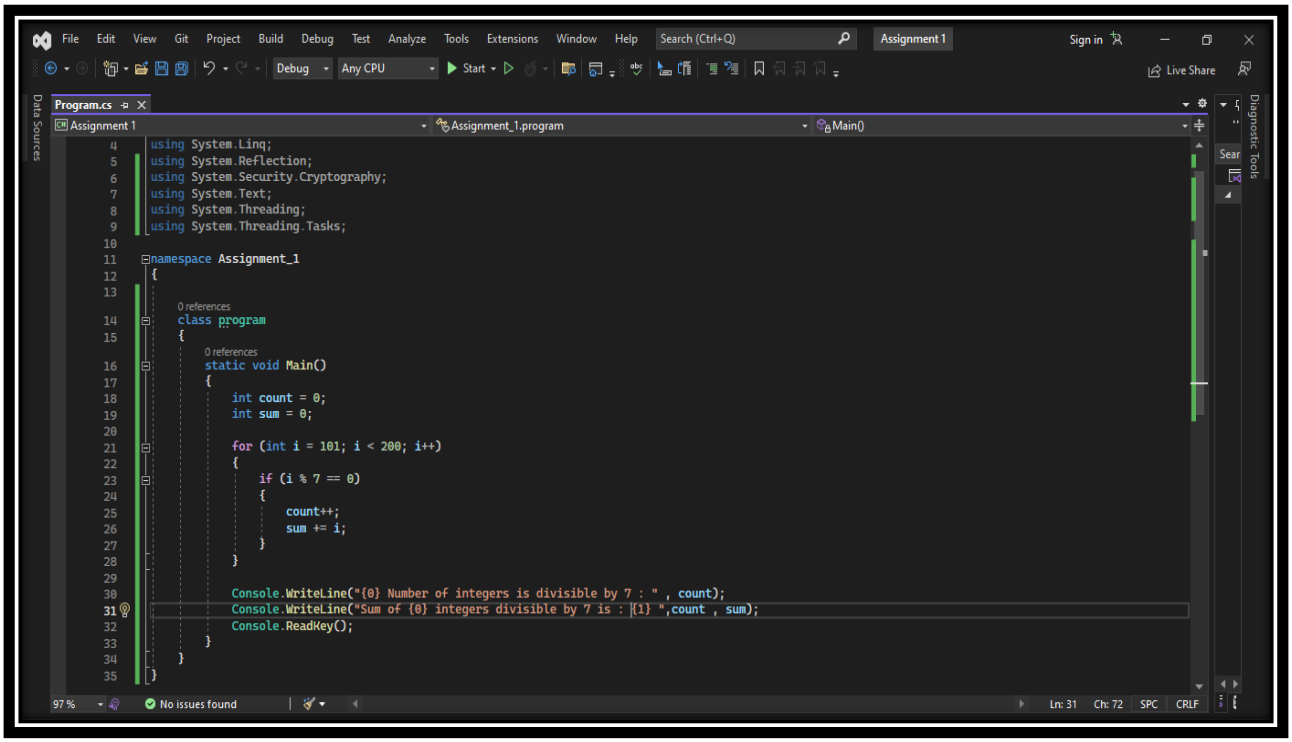
## Output:




## Program 9:

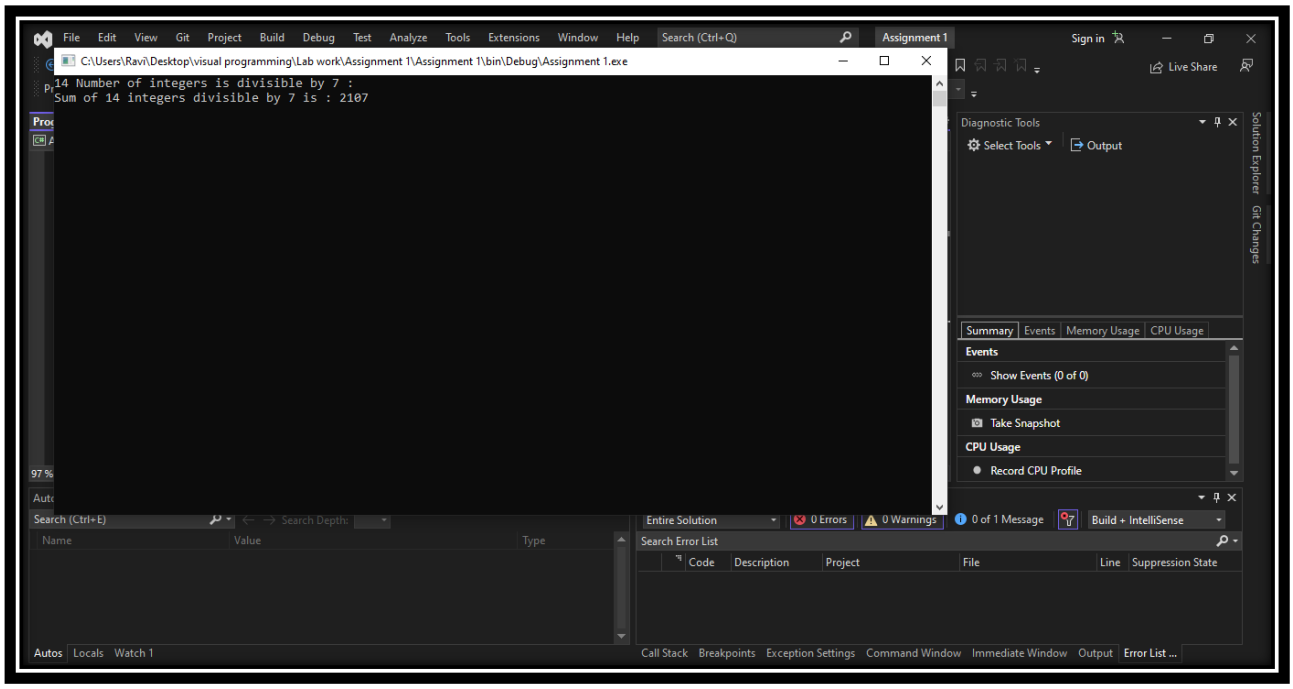
Write a program to find the number of and sum of all integers greater than 100 and less than 200 that are divisible by 7.

Solution:



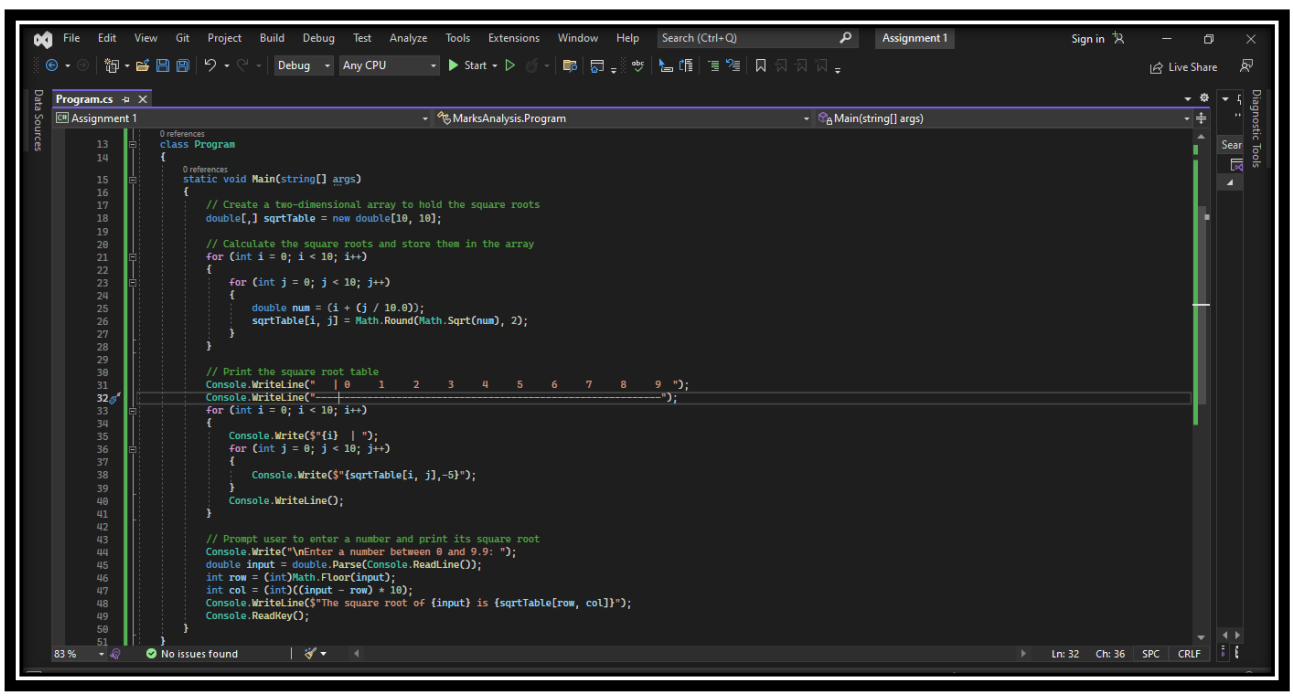
```
using System.Linq;
using System.Reflection;
using System.Security.Cryptography;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace Assignment_1
{

    class program
    {
        static void Main()
        {
            int count = 0;
            int sum = 0;

            for (int i = 101; i < 200; i++)
            {
                if (i % 7 == 0)
                {
                    count++;
                    sum += i;
                }
            }

            Console.WriteLine("{0} Number of integers is divisible by 7 : " , count);
            Console.WriteLine("Sum of {0} integers divisible by 7 is : {1} ",count , sum);
            Console.ReadKey();
        }
    }
}
```

## Output:



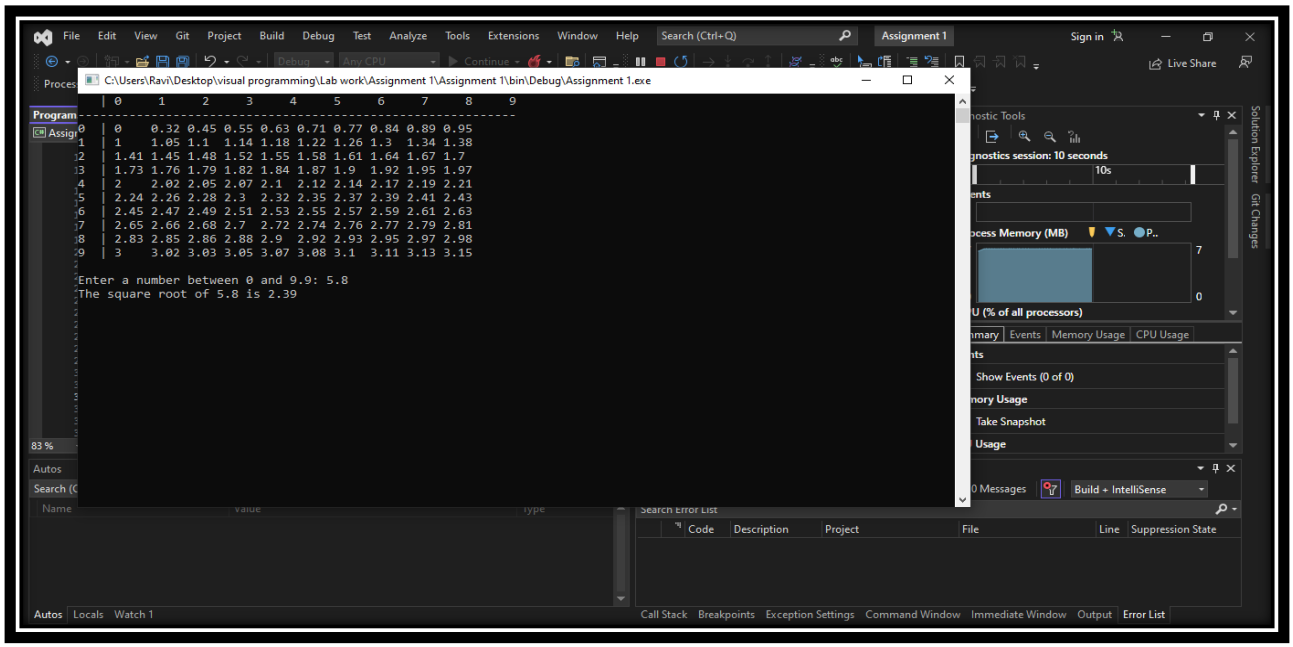

## Program 10:

Write a program to print a two-dimensional Square Root Table as shown below, to provide the square root of any number from 0 to 9.9. For example, the value x will give the square root of 3.2 and y the square root of 3.9.



```
class Program
{
    static void Main(string[] args)
    {
        // Create a two-dimensional array to hold the square roots
        double[,] sqrtTable = new double[10, 10];

        // Calculate the square roots and store them in the array
        for (int i = 0; i < 10; i++)
        {
            for (int j = 0; j < 10; j++)
            {
                double num = (i + (j / 10.0));
                sqrtTable[i, j] = Math.Round(Math.Sqrt(num), 2);
            }
        }

        // Print the square root table
        Console.WriteLine("   | 0    1    2    3    4    5    6    7    8    9 ");
        Console.WriteLine("---|-----------------------------------------------");
        for (int i = 0; i < 10; i++)
        {
            Console.Write($"{i}  | ");
            for (int j = 0; j < 10; j++)
            {
                Console.Write($"{sqrtTable[i, j],-5}");
            }
            Console.WriteLine();
        }

        // Prompt user to enter a number and print its square root
        Console.Write("\nEnter a number between 0 and 9.9: ");
        double input = double.Parse(Console.ReadLine());
        int row = (int)Math.Floor(input);
        int col = (int)((input - row) * 10);
        Console.WriteLine($"The square root of {input} is {sqrtTable[row, col]}");
        Console.ReadKey();
    }
}
```

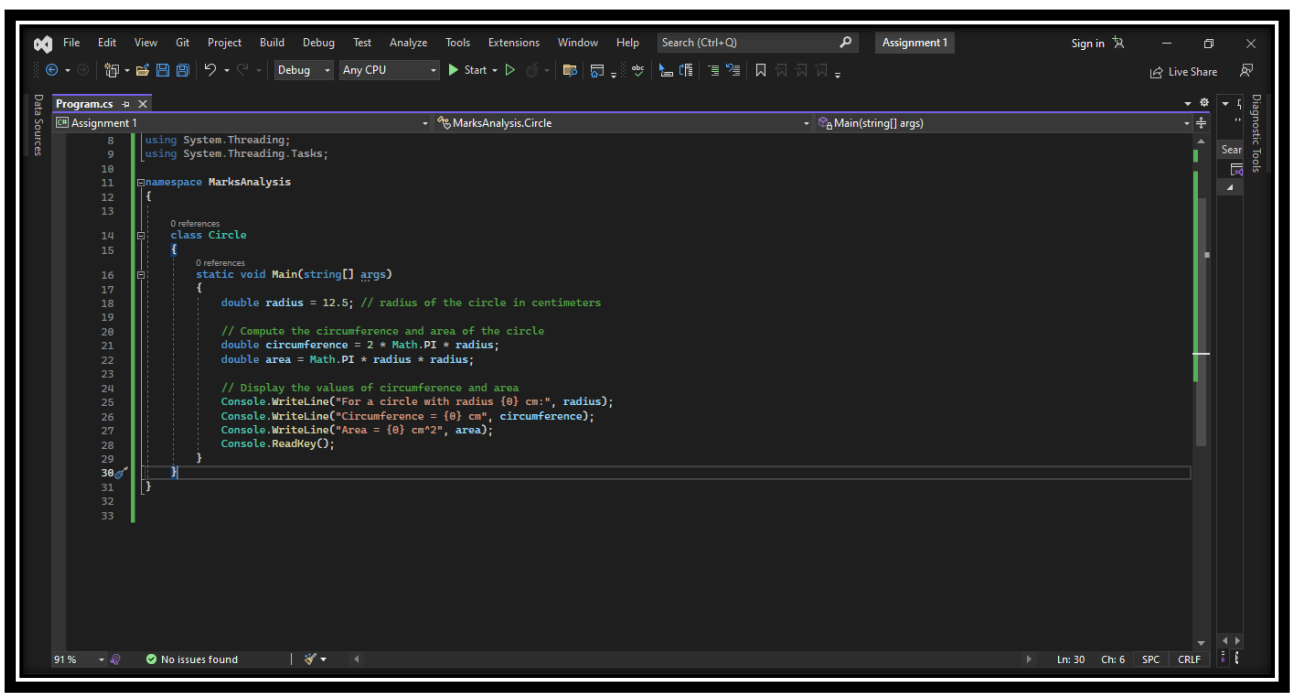
## Output:



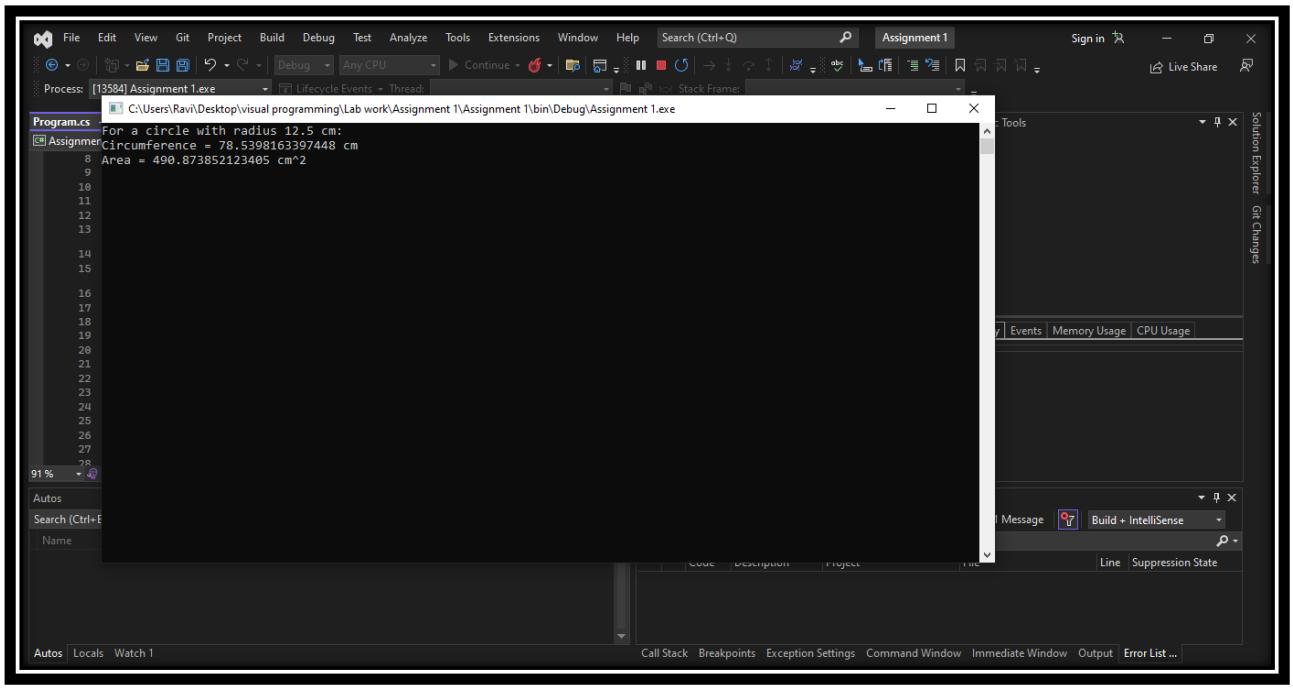

## Program 11:

Given the radius of the circle as 12.5 centimeters, write a program to compute its circumference and area and display their values.

## Solution:

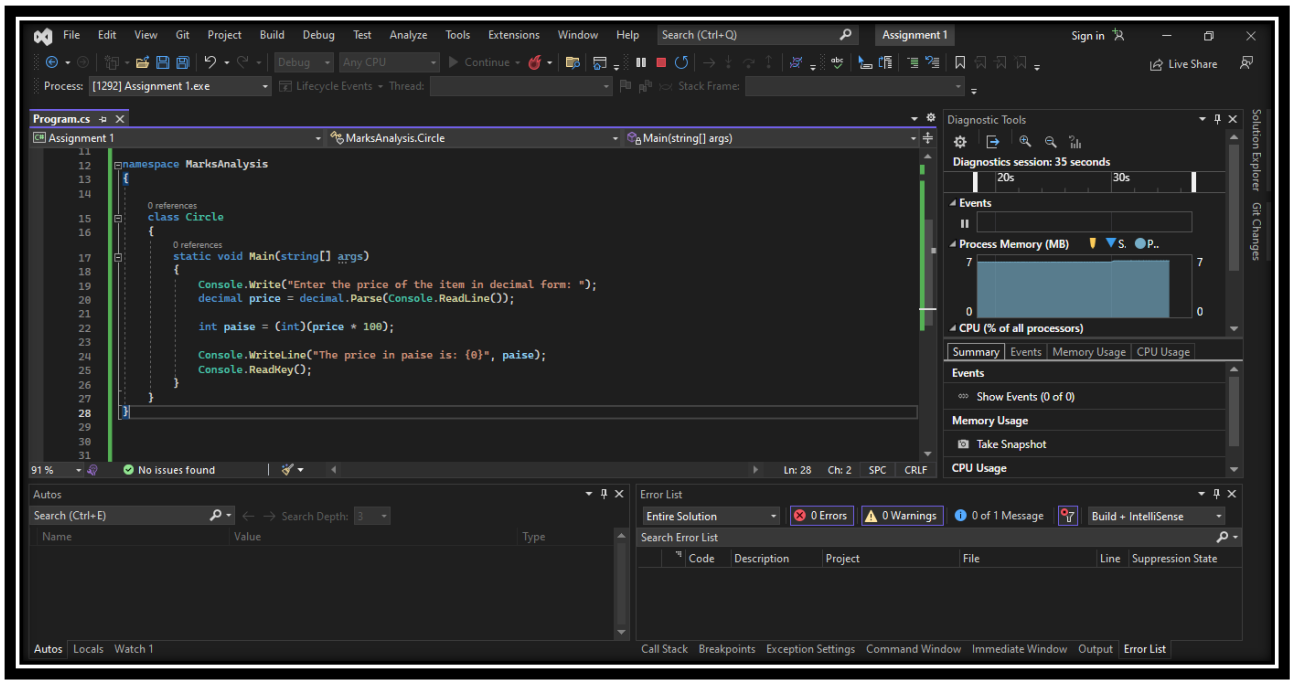


## Output:

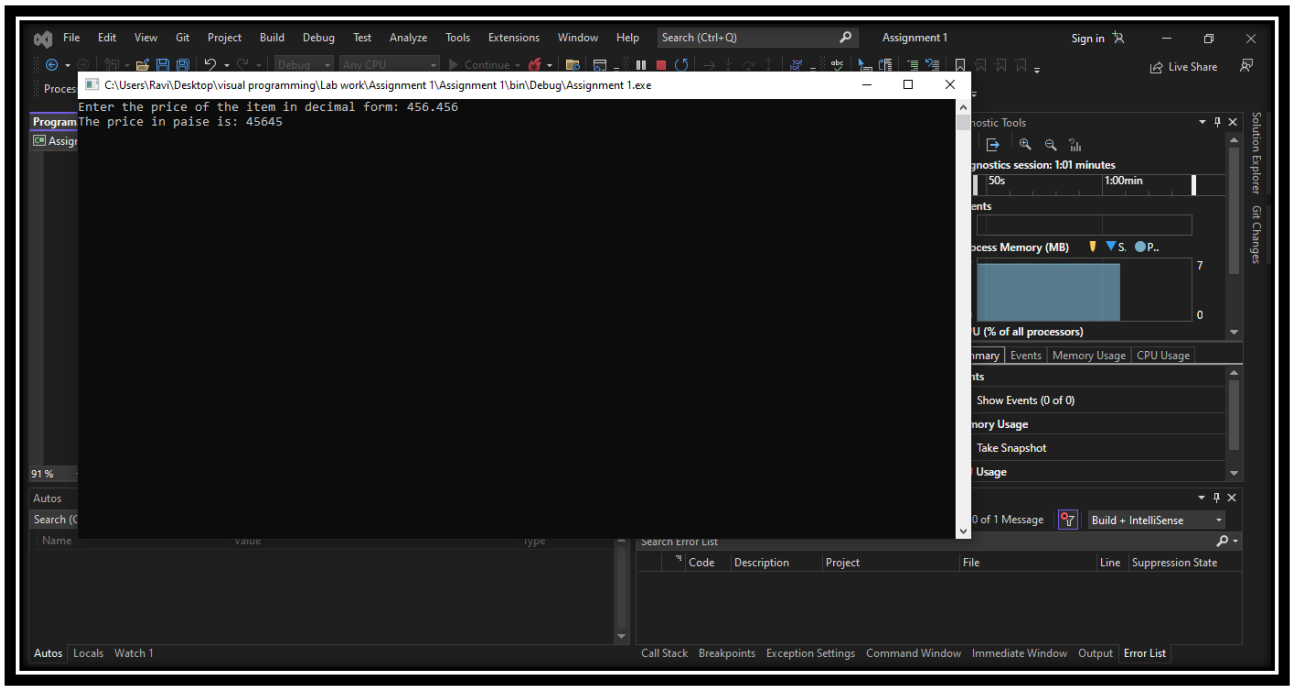



## Program 12:

Write a program to read the price of an item in decimal form (like 75.95) and print the output in paise (like 7595 paise).

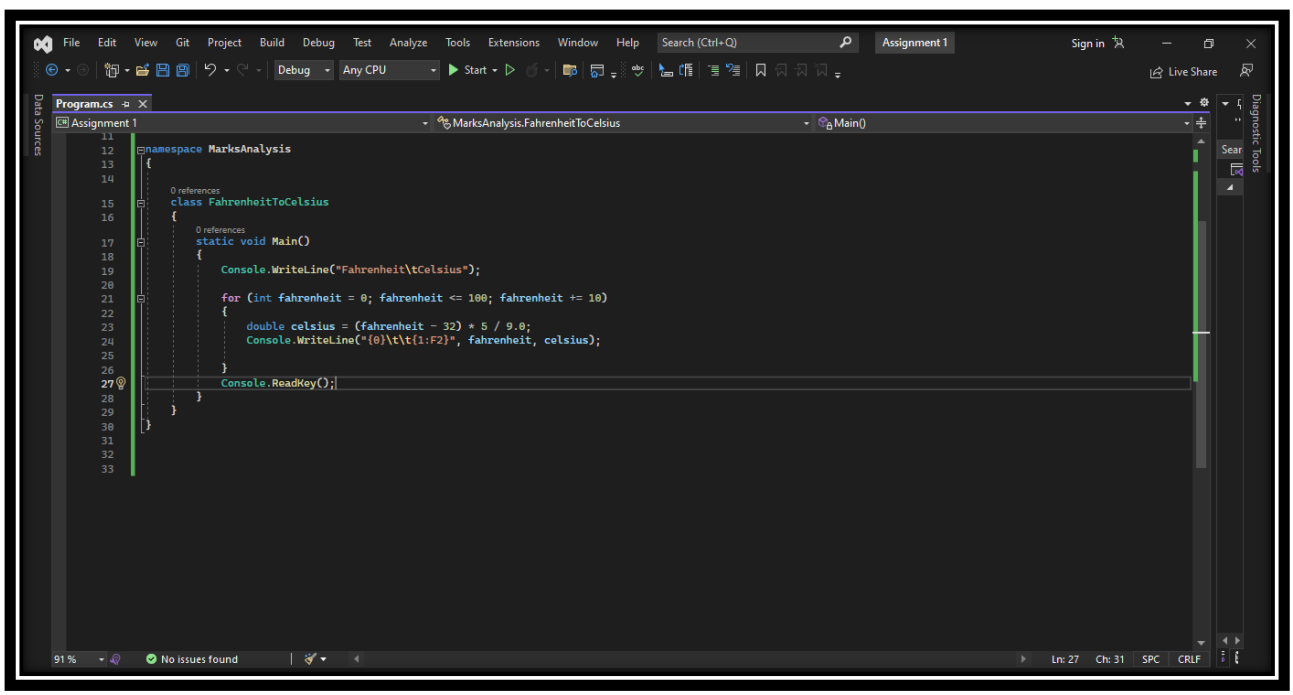


## Output:

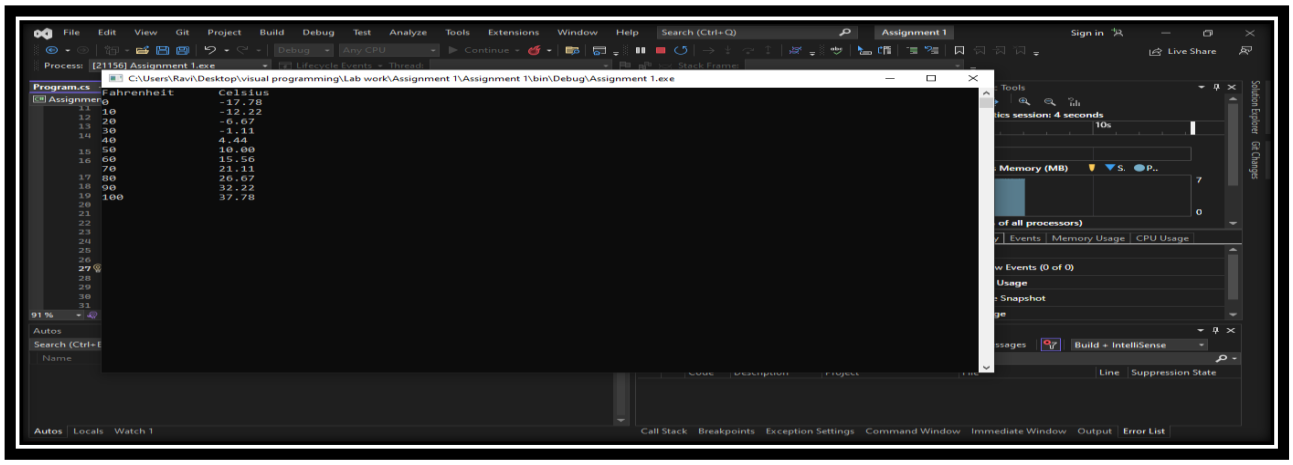



## Program 13:

Write a program to convert the given temperature in fahrenheit to celsius using the following conversion formula. C= F-321.8 and display the values in a tabular form.



## Output:



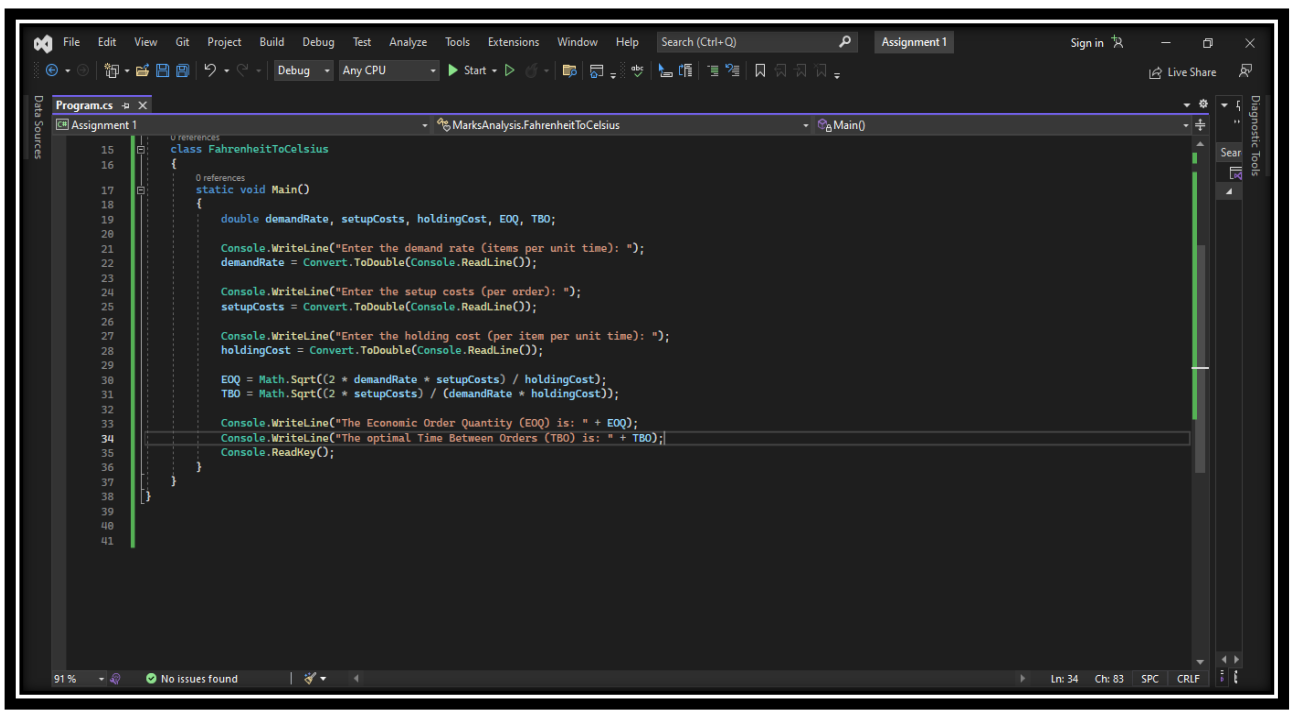

## Program 14:

In inventory management, the Economic Order Quantity for a single item is given by

EOQ= 2* demand rate* setup costs holding cost per item per unit time and the optimal Time Between Orders TBO= 2 *setup costs demand rate holding cost per item per unit time

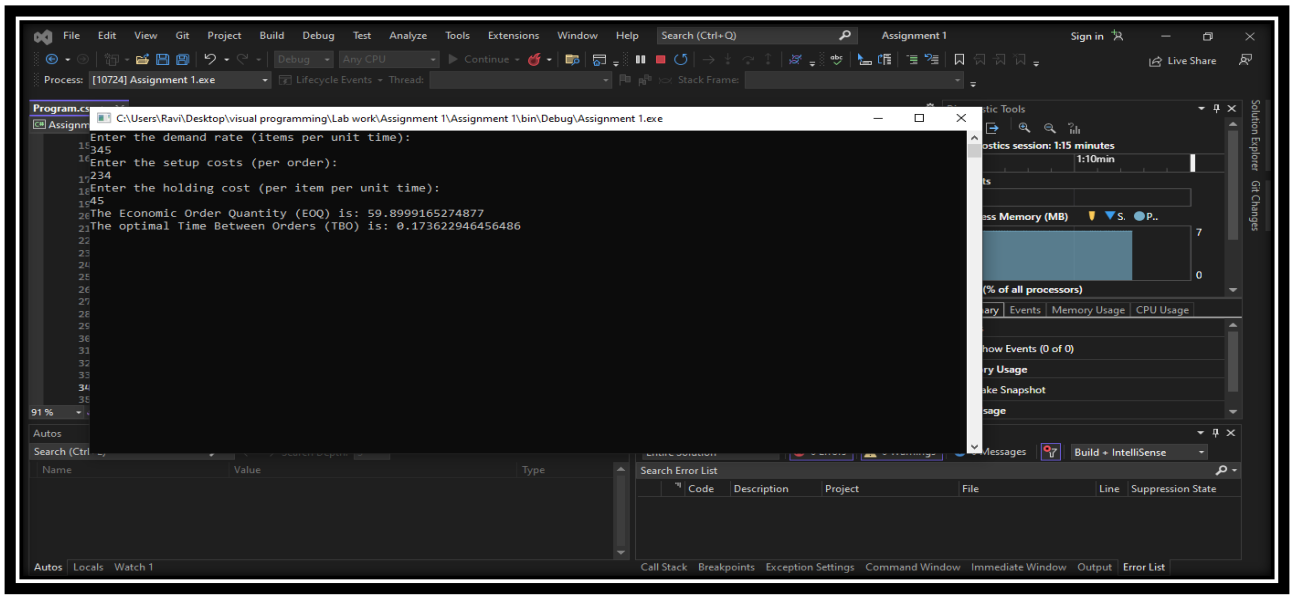
Write a program to computer EOQ and TBO, given demand rate (items per unit time), setup costs (per order), and the holding cost (per item per unit time).

## Solution:



```
class FahrenheitToCelsius
{
    static void Main()
    {
        double demandRate, setupCosts, holdingCost, EOQ, TBO;

        Console.WriteLine("Enter the demand rate (items per unit time): ");
        demandRate = Convert.ToDouble(Console.ReadLine());

        Console.WriteLine("Enter the setup costs (per order): ");
        setupCosts = Convert.ToDouble(Console.ReadLine());

        Console.WriteLine("Enter the holding cost (per item per unit time): ");
        holdingCost = Convert.ToDouble(Console.ReadLine());

        EOQ = Math.Sqrt((2 * demandRate * setupCosts) / holdingCost);
        TBO = Math.Sqrt((2 * setupCosts) / (demandRate * holdingCost));

        Console.WriteLine("The Economic Order Quantity (EOQ) is: " + EOQ);
        Console.WriteLine("The optimal Time Between Orders (TBO) is: " + TBO);
        Console.ReadKey();
    }
}
```

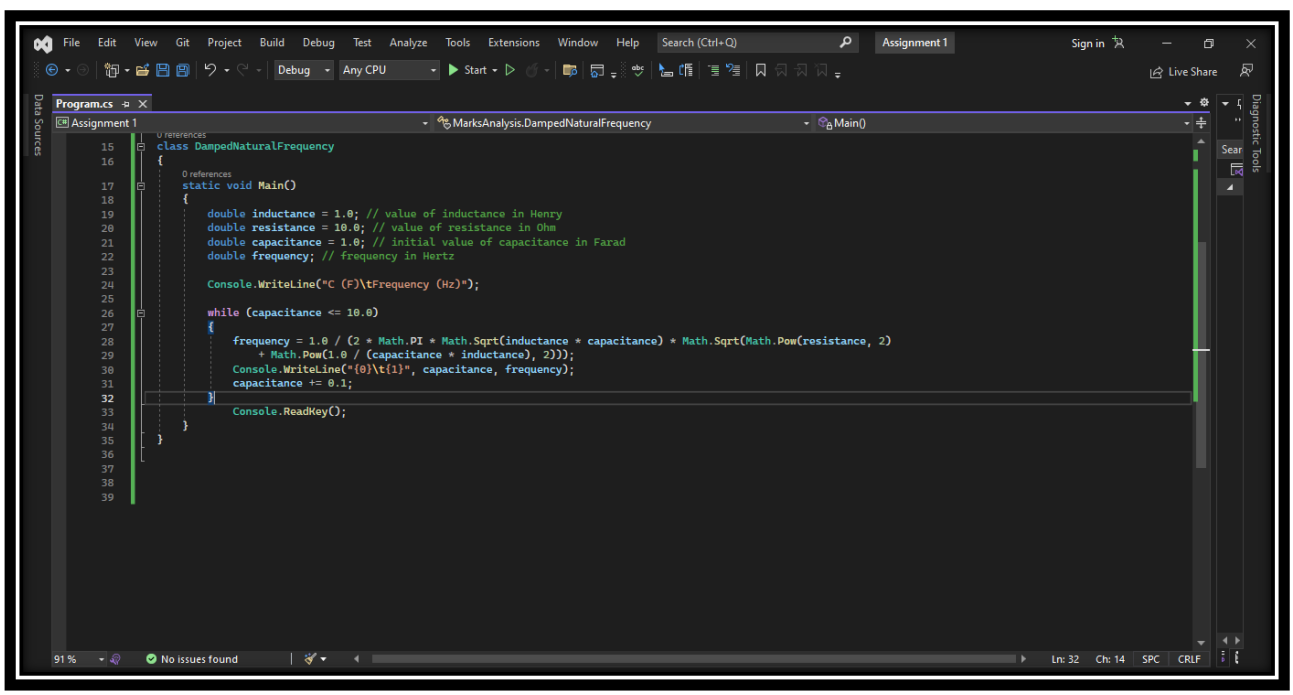
## Output:



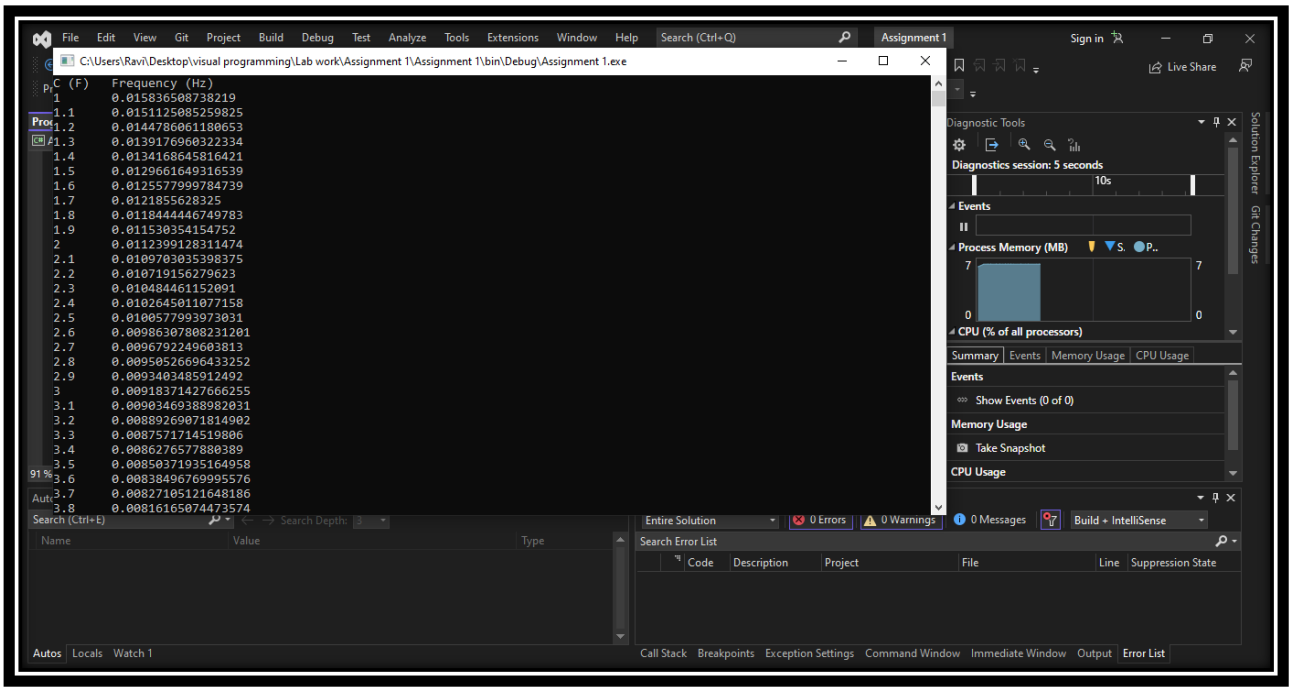

## Program 15:

For a certain electrical circuit with an inductance Land resistance R, the damped natural frequency is given by: Frequency=1 R2 VLC4C2 It is desired to study the variation of this frequency with C (capacitance).

Solution:



## Output:


C:\Users\Ravi\Desktop\visual programming\Lab work\Assignment 1\Assignment 1\bin\Debug\Assignment 1.exe
C (F) Frequency (Hz)
1 0.015836508738219
1.1 0.0151125085259825
1.2 0.0144786061180653
1.3 0.0139176960322334
1.4 0.0134168645816421
1.5 0.0129661649316539
1.6 0.0125577999784739
1.7 0.0121855628325
1.8 0.0118444446749783
1.9 0.011530354154752
2 0.0112399128311474
2.1 0.0109703035398375
2.2 0.010719156279623
2.3 0.010484461152091
2.4 0.0102645011077158
2.5 0.0100577993973031
2.6 0.00986307808231201
2.7 0.0096792249603813
2.8 0.00950526696433252
2.9 0.0093403485912492
3 0.00918371427666255
3.1 0.009034693888982031
3.2 0.00889269071814902
3.3 0.0087571714519806
3.4 0.0086276577880389
3.5 0.00850371935164958
3.6 0.00838496769995576
3.7 0.00827105121648186
3.8 0.00816165074473574