



DEV SANSKRITI
VISHWAVIDYALAYA

Practical file

On

C#.net

SUBMITTED TO:

Mr.Chandrashekhar Patel Sir

SUMBITTED BY

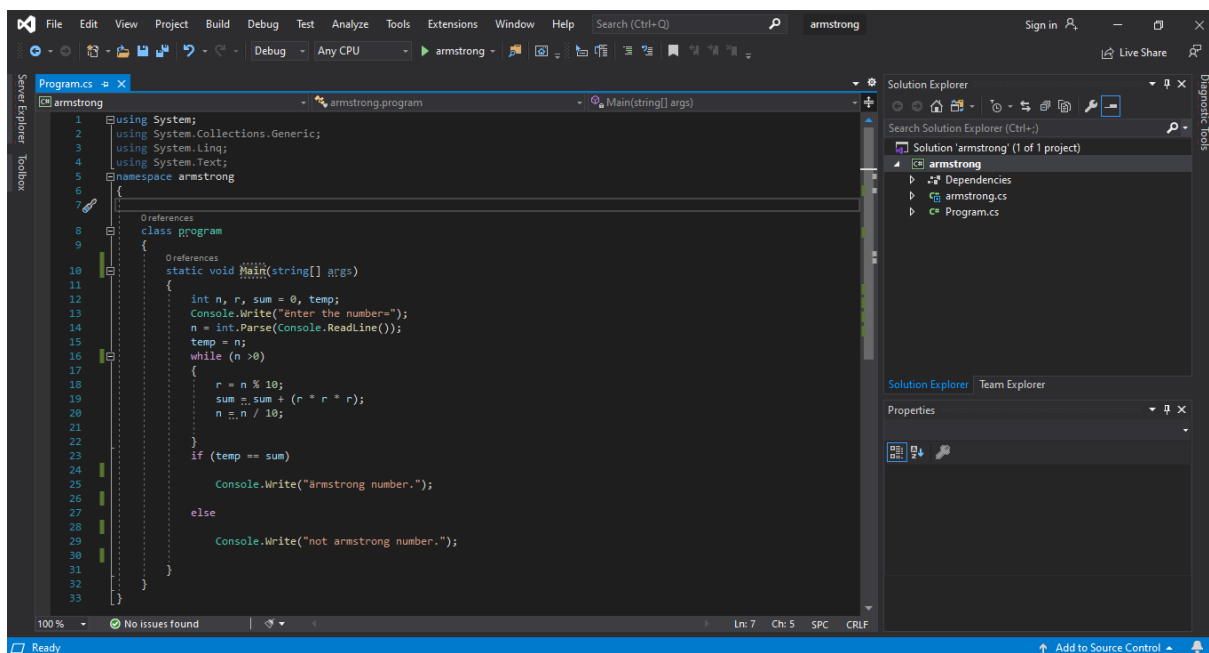
Aman Yadav (BCA5thsem)

Department of computer science

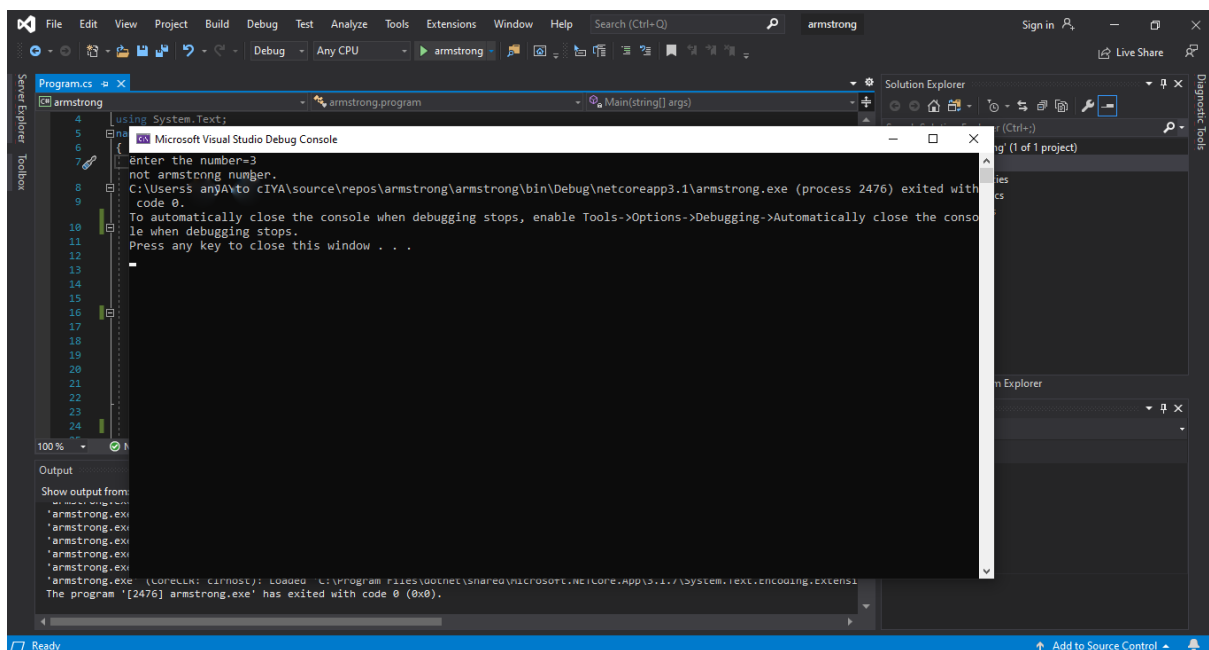
Sr.no	topics	Page no
1)	write a programme to print Armstrong number:-	1
2	write a programme for print factorial of a number:	2
3)	write a programme for find the GCD of 2 numbers:	3
4)	write a programme to check if a number is a prime number	4
5)	write a programme to print the fibonacci series :	5
6)	write a programme to print the half pyramid pattern:	6
7)	Write a programme to print the half pyramid pattern with numbers:	7
8)	Write a programme to print half pyramid inverse pattern:	8
9)	write a programme to print the pyramid pattern:	9
10)	Write a programme to print the inverse pyramid pattern:	10
11)	Write a programme to print the diamond pattern:	11
12)	Write a programme to print the pascal's triangle:	12
13)	Write a programme to compare 2 string without using string library function:	13
14)	Write a programme to count a total numbers of alphabets , digits and special characters in a string:	14
15)	Write a programme to copy one string to another string:	15
16)	Write a programme to find maximum occurring character in a string:	16
17)	Write a programme to check whether a given substring is present in the given string:	17
18)	Write a programme for abstraction:	18
19)	Write a programme for single inheritance:	19
20)	Write a programme for multi level inheritance	20

21)	Write a programme for multiple inheritance:	21-22
22)	Write a programme for method overloading:	23
23)	Write a programme for method overriding:	24
24)	Write a programme for interface:	25
25)	Write a programme for exception handling through try and catch:	26
26)	Write a programme for properties.	27
27)	Write a programme for threading.	28
28)	Write a programme to access data from database using ADO.net.	29
29)	Write a programme using namespace.	30-31

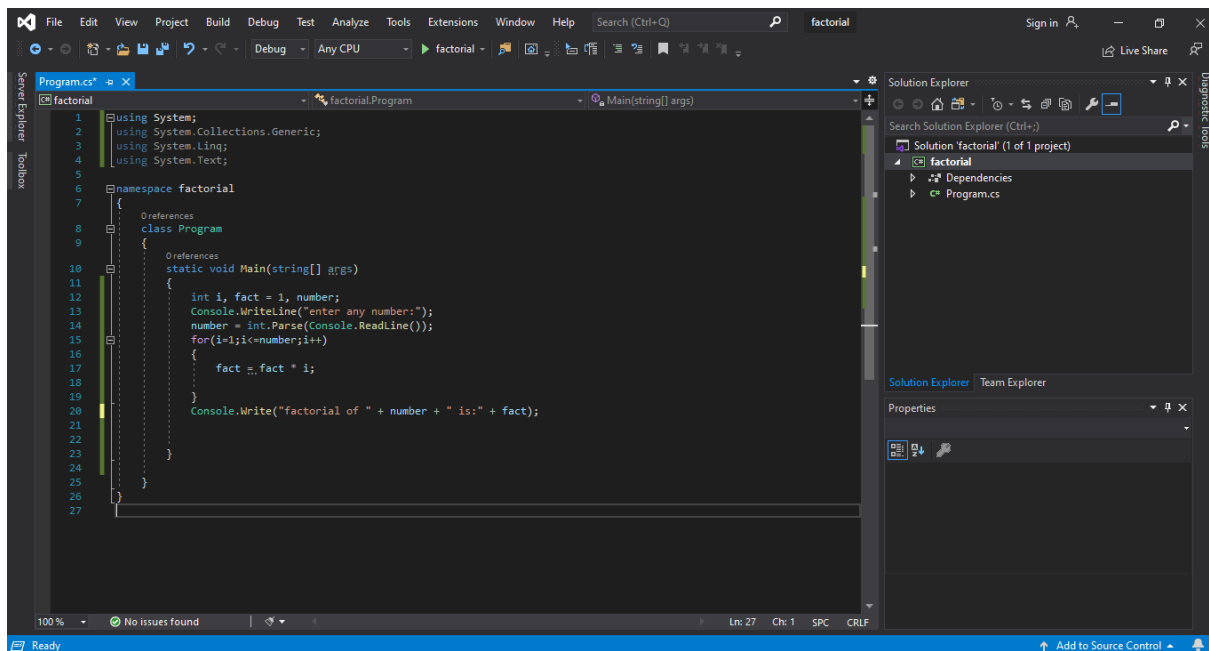
1] write a programme to print Armstrong number:-



Output:



2] write a programme for print factorial of a number:

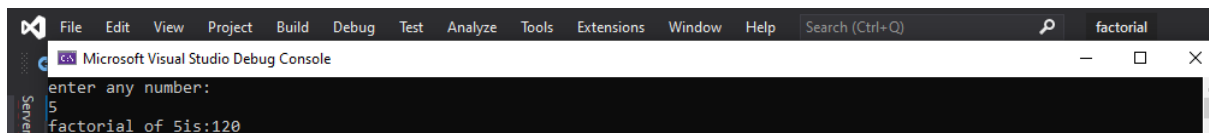


The screenshot shows the Visual Studio IDE with a C# console application named 'factorial'. The code in 'Program.cs' is as follows:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace factorial
7 {
8     class Program
9     {
10         static void Main(string[] args)
11         {
12             int i, fact = 1, number;
13             Console.WriteLine("enter any number:");
14             number = int.Parse(Console.ReadLine());
15             for(i=1; i<=number; i++)
16             {
17                 fact = fact * i;
18             }
19             Console.WriteLine("factorial of " + number + " is: " + fact);
20         }
21     }
22 }
23
24
25
26
27
```

The Solution Explorer on the right shows the project structure: 'factorial' (1 of 1 project) containing 'Dependencies' and 'Program.cs'. The status bar at the bottom indicates '100%' zoom, 'No issues found', and 'Ln: 27, Cln: 1, SPC, CRLF'.

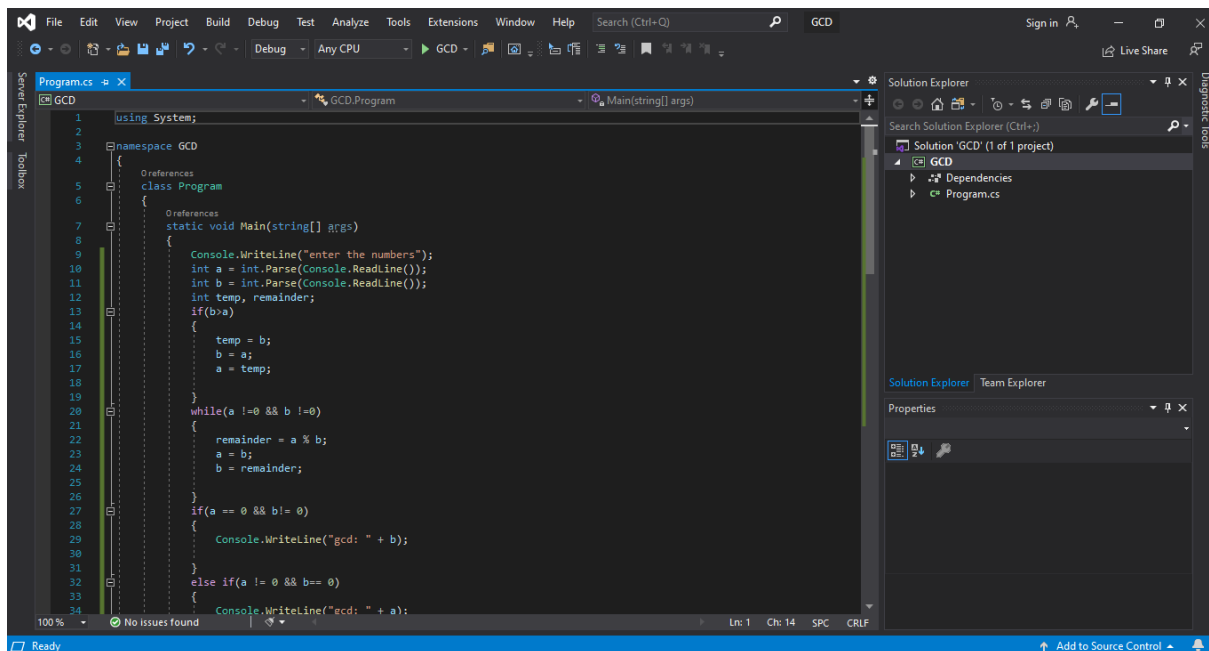
Output:



The screenshot shows the 'Microsoft Visual Studio Debug Console' window. It displays the program's execution output:

```
enter any number:
5
factorial of 5 is:120
```

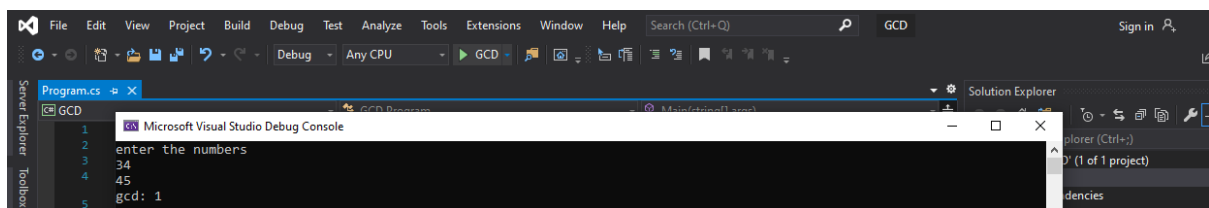
3] write a programme for find the GCD of 2 numbers:



The screenshot shows the Visual Studio IDE with a C# program named 'Program.cs' open. The code implements a GCD algorithm using the Euclidean method. The 'Solution Explorer' on the right shows the project structure with 'GCD' as the project and 'Program.cs' as the source file. The status bar at the bottom indicates 'Ready' and 'No issues found'.

```
1 using System;
2
3 namespace GCD
4 {
5     References
6     class Program
7     {
8         References
9         static void Main(string[] args)
10         {
11             Console.WriteLine("enter the numbers");
12             int a = int.Parse(Console.ReadLine());
13             int b = int.Parse(Console.ReadLine());
14             int temp, remainder;
15             if(b>a)
16             {
17                 temp = b;
18                 b = a;
19                 a = temp;
20             }
21             while(a != 0 && b != 0)
22             {
23                 remainder = a % b;
24                 a = b;
25                 b = remainder;
26             }
27             if(a == 0 && b != 0)
28             {
29                 Console.WriteLine("gcd: " + b);
30             }
31             else if(a != 0 && b == 0)
32             {
33                 Console.WriteLine("gcd: " + a);
34             }
35         }
36     }
37 }
```

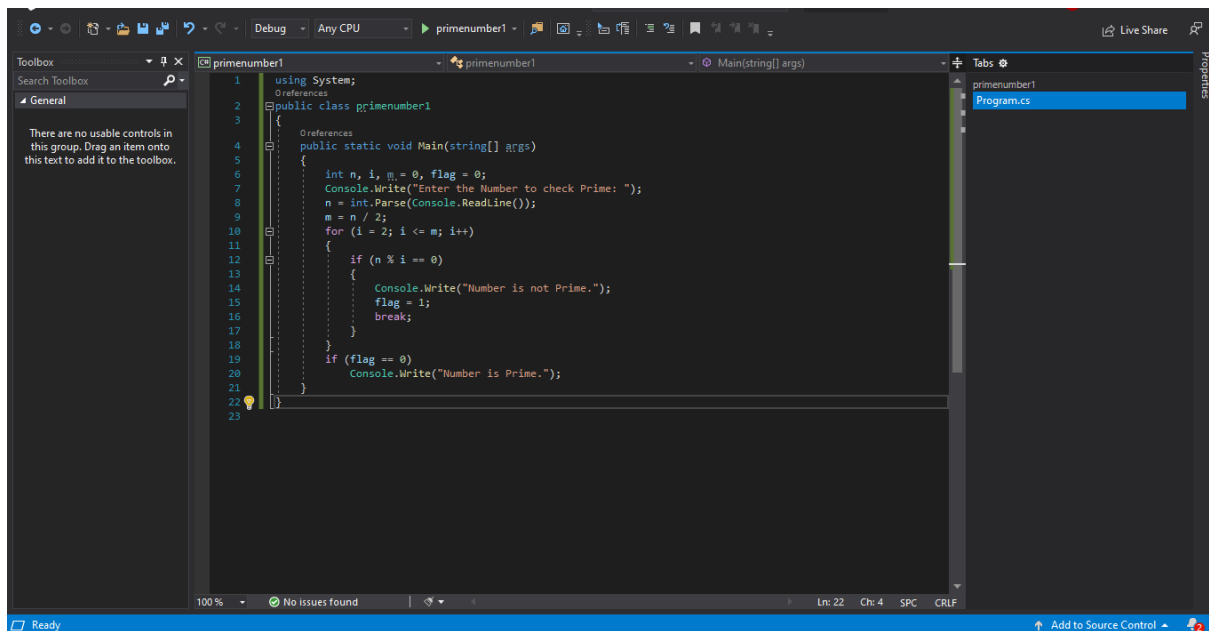
Output:



The screenshot shows the Visual Studio IDE with the 'Microsoft Visual Studio Debug Console' window open. The console displays the output of the program, which prompts the user to 'enter the numbers' and then shows the calculated GCD of 34 and 45, which is 1.

```
1 enter the numbers
2 34
3 45
4 gcd: 1
```

4] write a programme to check if a number is a prime number

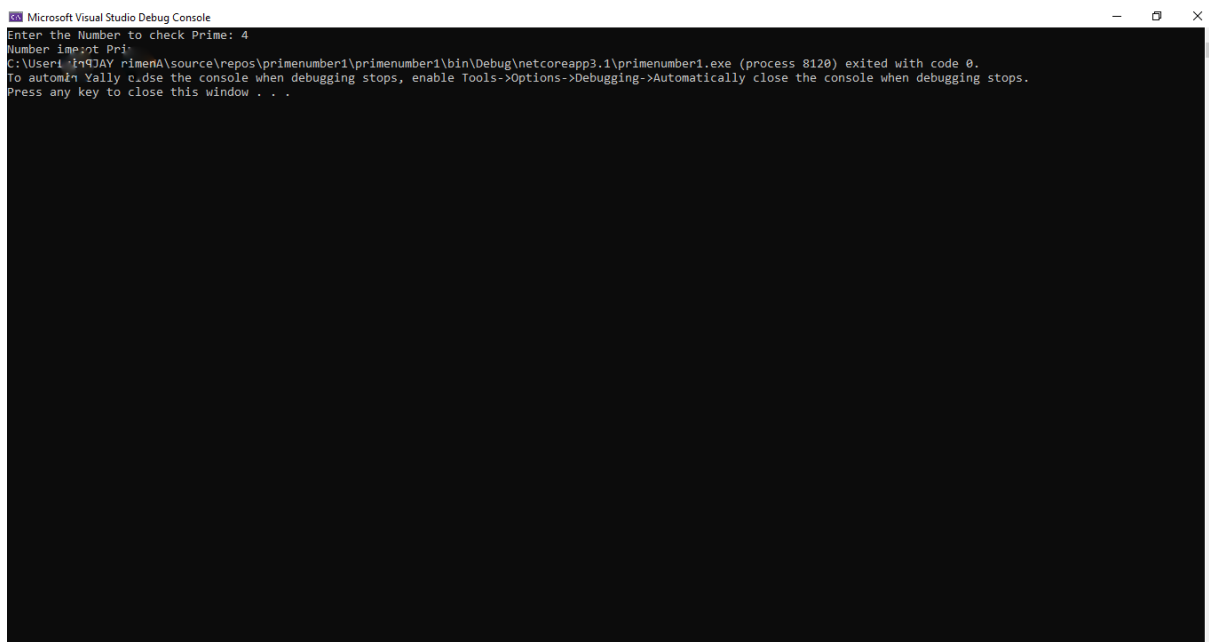


The screenshot shows the Visual Studio IDE with a C# project named 'primenumber1'. The code in 'Program.cs' is as follows:

```
1 using System;
2
3 public class primenumber1
4 {
5     public static void Main(string[] args)
6     {
7         int n, i, m = 0, flag = 0;
8         Console.Write("Enter the Number to check Prime: ");
9         n = int.Parse(Console.ReadLine());
10        m = n / 2;
11        for (i = 2; i <= m; i++)
12        {
13            if (n % i == 0)
14            {
15                Console.Write("Number is not Prime.");
16                flag = 1;
17                break;
18            }
19        }
20        if (flag == 0)
21            Console.Write("Number is Prime.");
22    }
23 }
```

The status bar at the bottom indicates '100%' zoom, 'No issues found', and the cursor is at line 22, column 4.

Output:



The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
Enter the Number to check Prime: 4
Number is not Pri-
C:\Users\rajay\source\repos\primenumber1\primenumber1\bin\Debug\netcoreapp3.1\primenumber1.exe (process 8120) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

5] write a programme to print the fibonacci series :

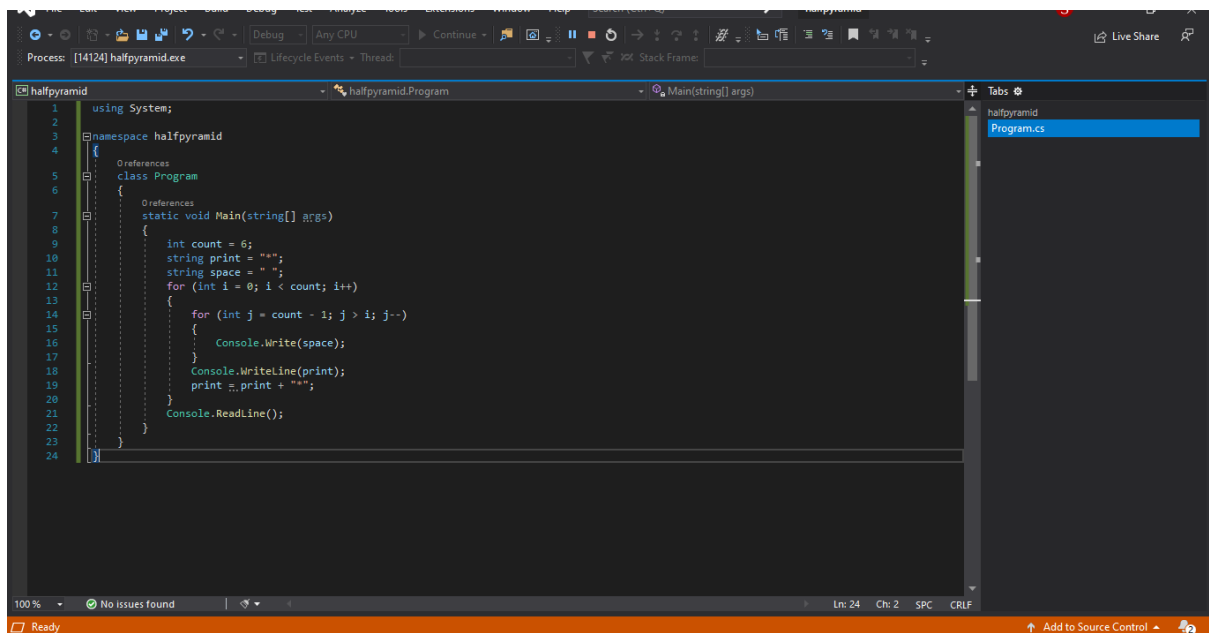
The screenshot displays the Visual Studio IDE with a C# project named 'fibonacci'. The main editor window shows the source code for 'Program.cs', which includes the following code:

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace fibonacci
7 {
8     References
9     class Program
10     {
11         References
12         static void Main(string[] args)
13         {
14             int n1 = 0, n2 = 1, n3, i, number;
15
16             Console.WriteLine("enter the numbers of elements");
17             number = int.Parse(Console.ReadLine());
18             Console.WriteLine(n1 + " " + n2 + " ");
19             for (i = 2; i < number; ++i)
20             {
21                 n3 = n1 + n2;
22                 Console.WriteLine(n3 + " ");
23                 n1 = n2;
24                 n2 = n3;
25             }
26         }
27     }
28 }
```

The right sidebar shows the 'Solution Explorer' with the project 'fibonacci' expanded, showing 'Dependencies' and 'Program.cs'. The bottom status bar indicates '100%' zoom, 'No issues found', and the current cursor position is 'Ln: 14, Ch: 61, SPC, CRLF'.

Output:

6] write a programme to print the half pyramid pattern:



The screenshot shows a Visual Studio IDE with a C# program named 'halfpyramid'. The code is as follows:

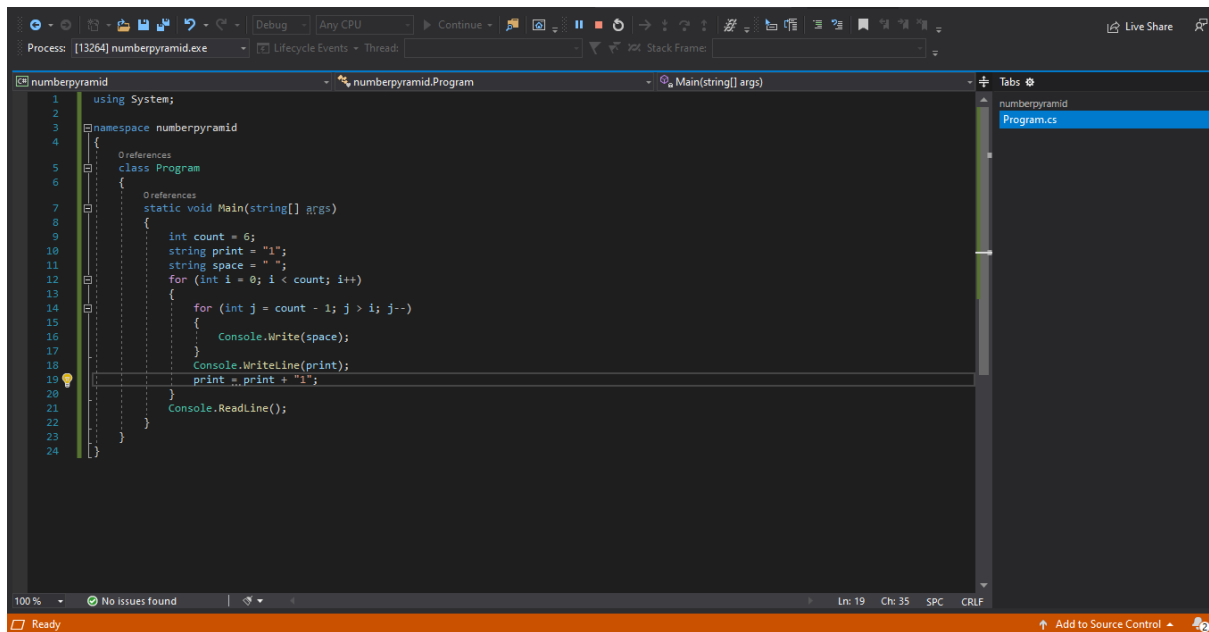
```
1 using System;
2
3 namespace halfpyramid
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             int count = 6;
10            string print = "";
11            string space = " ";
12            for (int i = 0; i < count; i++)
13            {
14                for (int j = count - 1; j > i; j--)
15                {
16                    Console.Write(space);
17                }
18                Console.WriteLine(print);
19                print += print + " ";
20            }
21            Console.ReadLine();
22        }
23    }
24 }
```

The right sidebar shows the 'Program.cs' file. The status bar at the bottom indicates '100%' zoom, 'No issues found', and 'Ln: 24, Ch: 2, SPC, CRLF'.

Output:



7] Write a programme to print the half pyramid pattern with numbers:



The screenshot shows the Visual Studio IDE with a C# program named 'numberpyramid'. The code is as follows:

```
1 using System;
2
3 namespace numberpyramid
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             int count = 6;
10            string print = "1";
11            string space = " ";
12            for (int i = 0; i < count; i++)
13            {
14                for (int j = count - 1; j > i; j--)
15                {
16                    Console.Write(space);
17                }
18                Console.WriteLine(print);
19                print = print + "1";
20            }
21            Console.ReadLine();
22        }
23    }
24 }
```

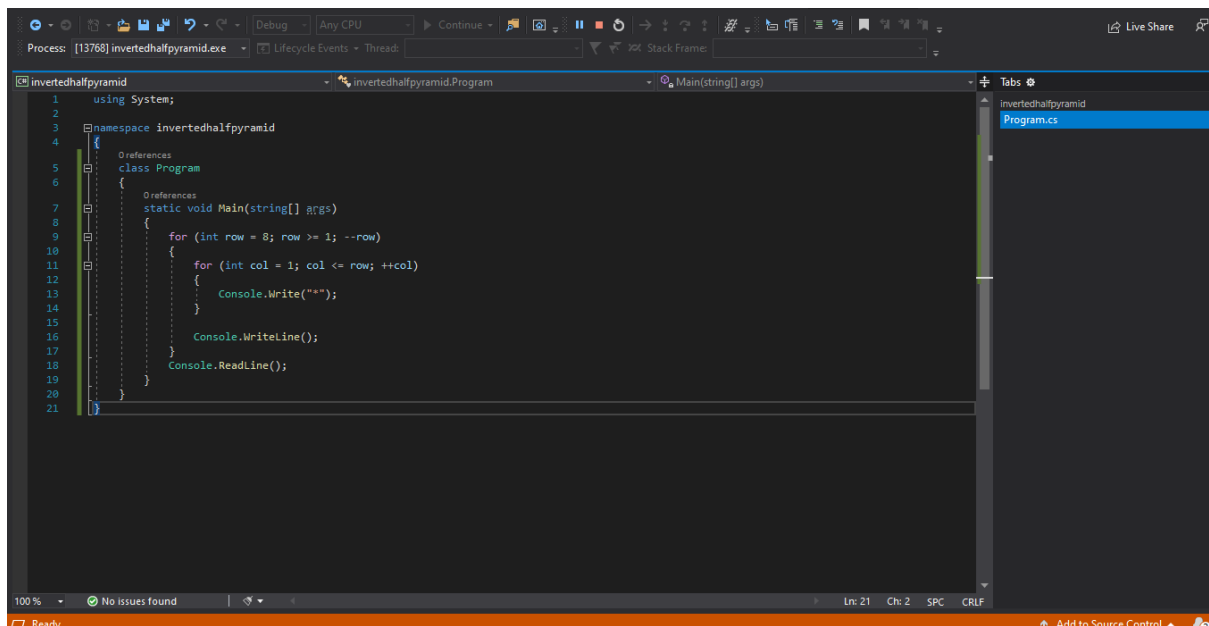
Output:



The screenshot shows the console output of the program, which is a half pyramid pattern of numbers:

```
1
11
111
1111
11111
111111
```

8] Write a programme to print half pyramid inverse pattern:



The screenshot shows a Visual Studio IDE with a C# program. The code is as follows:

```
1 using System;
2
3 namespace invertedhalfpyramid
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             for (int row = 8; row >= 1; --row)
10             {
11                 for (int col = 1; col <= row; ++col)
12                 {
13                     Console.Write("*");
14                 }
15                 Console.WriteLine();
16             }
17             Console.ReadLine();
18         }
19     }
20 }
21
```

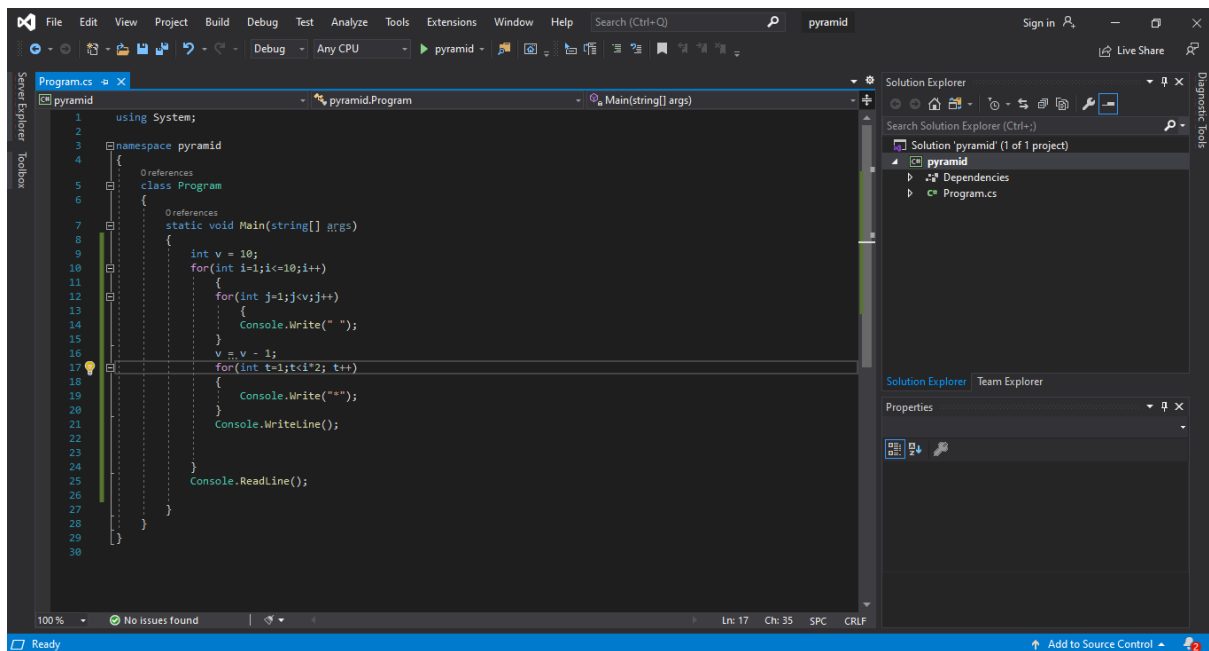
Output:



The screenshot shows the console output of the program, which is an inverted half pyramid pattern of asterisks:

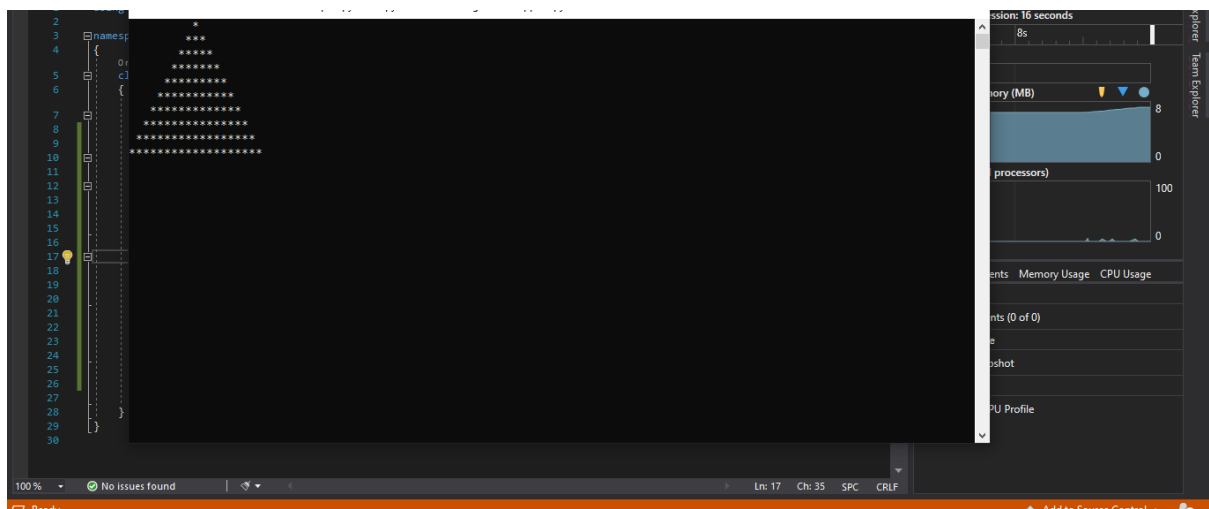
```
*****
****
***
**
*
```

9] write a programme to print the pyramid pattern:



```
1 using System;
2
3 namespace pyramid
4 {
5     References
6     class Program
7     {
8         References
9         static void Main(string[] args)
10         {
11             int v = 10;
12             for(int i=1;i<=10;i++)
13             {
14                 for(int j=1;j<=v;j++)
15                 {
16                     Console.Write(" ");
17                 }
18                 v = v - 1;
19                 for(int t=1;t<=i*t; t++)
20                 {
21                     Console.Write("");
22                 }
23                 Console.WriteLine();
24             }
25             Console.ReadLine();
26         }
27     }
28 }
29
30
```

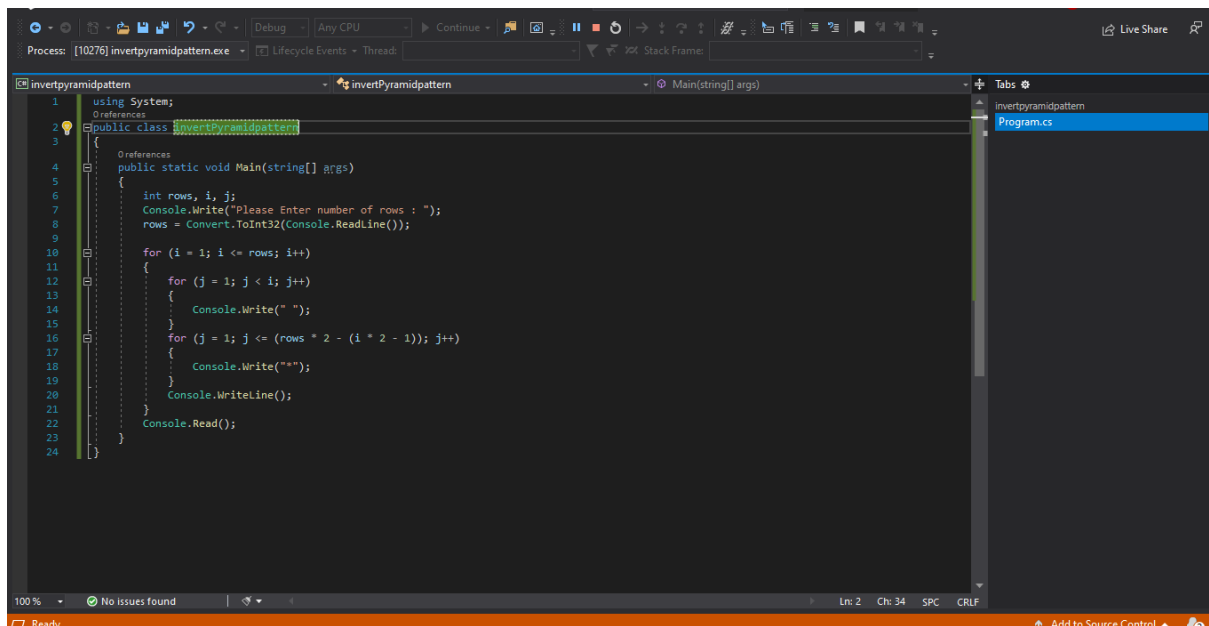
Output:



```
2 namespace
3 {
4     References
5     class
6     {
7         References
8         static void Main(string[] args)
9         {
10             int v = 10;
11             for(int i=1;i<=10;i++)
12             {
13                 for(int j=1;j<=v;j++)
14                 {
15                     Console.Write(" ");
16                 }
17                 v = v - 1;
18                 for(int t=1;t<=i*t; t++)
19                 {
20                     Console.Write("");
21                 }
22                 Console.WriteLine();
23             }
24             Console.ReadLine();
25         }
26     }
27 }
28
29
30
```

Performance Profiler window showing metrics for the execution of the program. The window displays a timeline of the execution, with a total duration of 16 seconds. The metrics shown include Memory Usage (MB), CPU Usage, and I/O Operations. The Memory Usage graph shows a peak of approximately 8 MB. The CPU Usage graph shows a peak of approximately 100%.

10] Write a programme to print the inverse pyramid pattern:



The screenshot shows a C# program in Visual Studio. The code defines a class `invertPyramidpattern` with a `Main` method. The `Main` method prompts the user to enter the number of rows, reads the input, and then uses nested loops to print an inverse pyramid pattern. The pattern consists of 6 rows, with each row containing a series of asterisks followed by spaces. The first row has 6 asterisks, the second has 5, and so on, down to the sixth row which has 1 asterisk. The program also includes a `Console.ReadLine()` at the end to wait for user input.

```
1 using System;
2 public class invertPyramidpattern
3 {
4     public static void Main(string[] args)
5     {
6         int rows, i, j;
7         Console.WriteLine("Please Enter number of rows : ");
8         rows = Convert.ToInt32(Console.ReadLine());
9
10        for (i = 1; i <= rows; i++)
11        {
12            for (j = 1; j < i; j++)
13            {
14                Console.Write(" ");
15            }
16            for (j = 1; j <= (rows * 2 - (i * 2 - 1)); j++)
17            {
18                Console.Write("*");
19            }
20            Console.WriteLine();
21        }
22        Console.ReadLine();
23    }
24 }
```

Output:



The screenshot shows the output of the program. It starts with the prompt "Please Enter number of rows : 6". The user has entered 6, and the program has printed an inverse pyramid pattern. The pattern consists of 6 rows, with each row containing a series of asterisks followed by spaces. The first row has 6 asterisks, the second has 5, and so on, down to the sixth row which has 1 asterisk. The pattern is as follows:

```
*****
*****
*****
***
**
*
```

11] Write a programme to print the diamond pattern:

The screenshot shows the Visual Studio IDE with the 'diamondpattern' project open. The 'Exercise31.cs' file is active, displaying the following C# code:

```

1  using System;
2  public class Exercise31
3  {
4      public static void Main()
5      {
6          int i, j, r;
7
8          Console.WriteLine("Input number of rows :");
9          r = Convert.ToInt32(Console.ReadLine());
10         for (i = 0; i <= r; i++)
11         {
12             for (j = 1; j <= r - i; j++)
13                 Console.Write(" ");
14             for (j = 1; j <= 2 * i - 1; j++)
15                 Console.Write("*");
16             Console.WriteLine("\n");
17         }
18         for (i = r - 1; i >= 1; i--)
19         {
20             for (j = 1; j <= r - i; j++)
21                 Console.Write(" ");
22             for (j = 1; j <= 2 * i - 1; j++)
23                 Console.Write("*");
24             Console.WriteLine("\n");
25         }
26     }
27 }

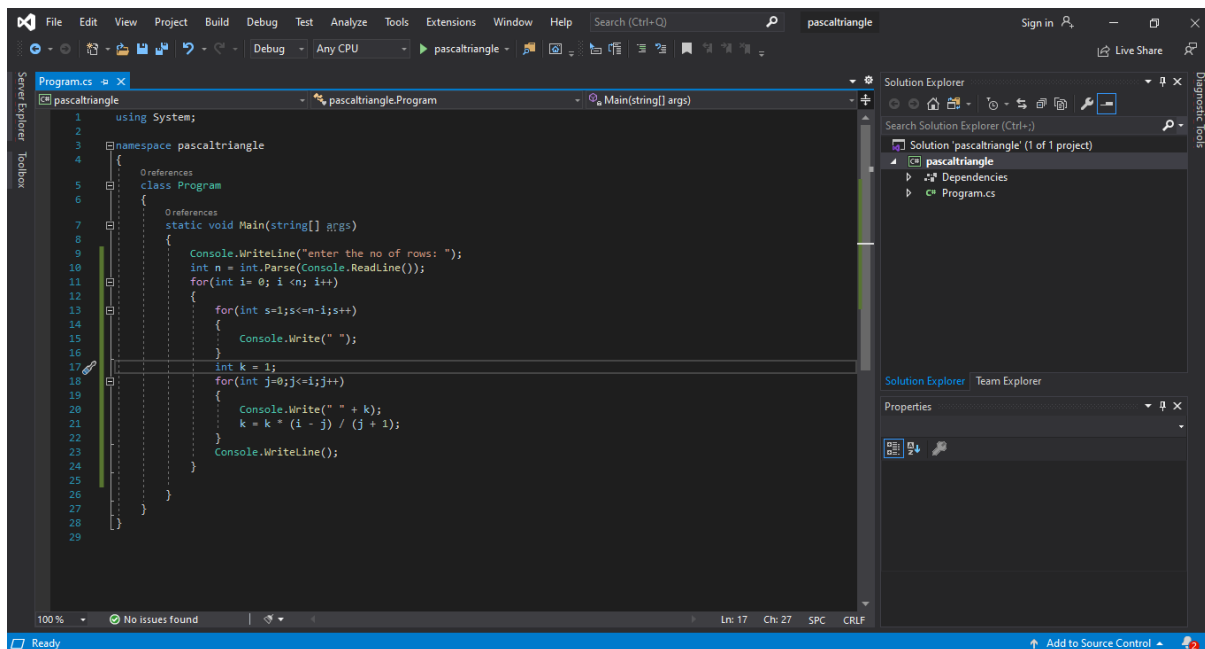
```

The interface includes a toolbar at the top, a search box on the left, a 'General' tab, and a tabs panel on the right showing 'diamondpattern' and 'Program.cs'. The status bar at the bottom indicates '100%' zoom, 'No issues found', and the current line is 'Ln: 8, Ch: 9, SPC, CRLF'.

Output:

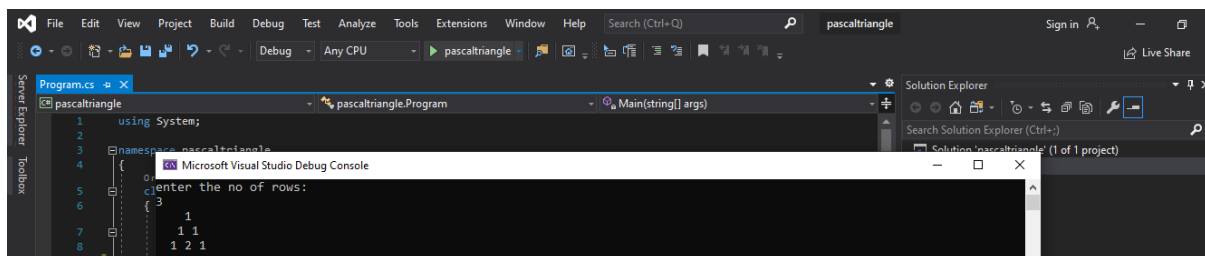
[illegible]

12] Write a programme to print the pascal's triangle:



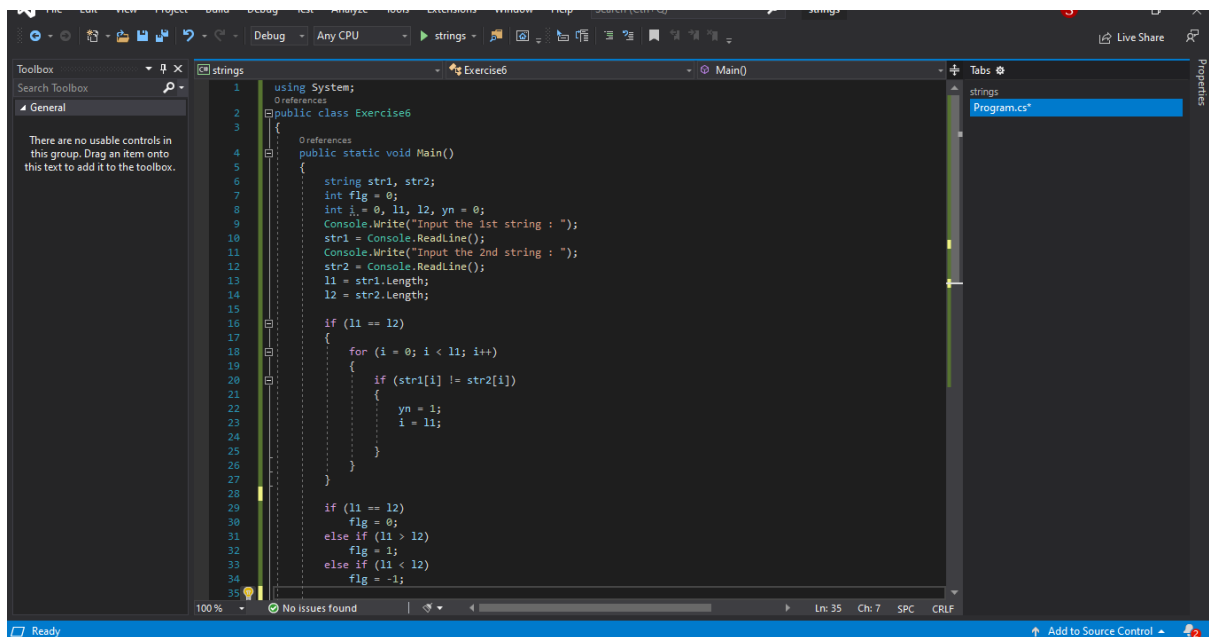
```
1 using System;
2
3 namespace pascaltriangle
4 {
5     References
6     class Program
7     {
8         References
9         static void Main(string[] args)
10         {
11             Console.WriteLine("enter the no of rows: ");
12             int n = int.Parse(Console.ReadLine());
13             for(int i = 0; i < n; i++)
14             {
15                 for(int s = i; s <= n - i; s++)
16                 {
17                     Console.Write(" ");
18                 }
19                 int k = 1;
20                 for(int j = 0; j <= i; j++)
21                 {
22                     Console.Write(" " + k);
23                     k = k * (i - j) / (j + 1);
24                 }
25                 Console.WriteLine();
26             }
27         }
28     }
29 }
```

Output:

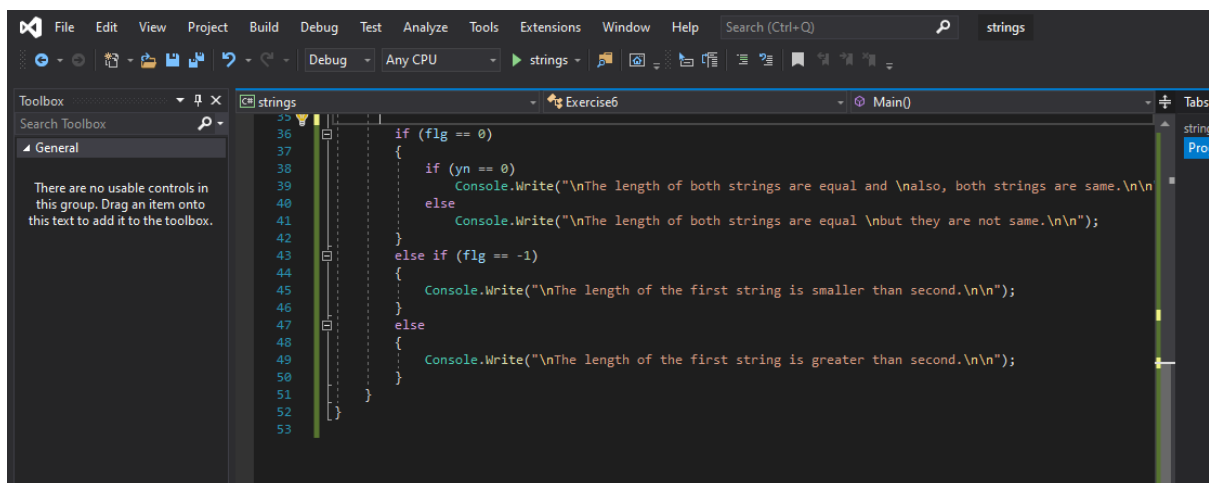


```
1 using System;
2
3 namespace pascaltriangle
4 {
5     References
6     class Program
7     {
8         References
9         static void Main(string[] args)
10         {
11             Console.WriteLine("enter the no of rows: ");
12             int n = 3;
13             for(int i = 0; i < n; i++)
14             {
15                 for(int s = i; s <= n - i; s++)
16                 {
17                     Console.Write(" ");
18                 }
19                 int k = 1;
20                 for(int j = 0; j <= i; j++)
21                 {
22                     Console.Write(" " + k);
23                     k = k * (i - j) / (j + 1);
24                 }
25                 Console.WriteLine();
26             }
27         }
28     }
29 }
```

13] Write a programme to compare 2 string without using string library function:



```
1 using System;
2 public class Exercise6
3 {
4     References
5     public static void Main()
6     {
7         string str1, str2;
8         int flg = 0;
9         int i, l1, l2, yn = 0;
10        Console.WriteLine("Input the 1st string : ");
11        str1 = Console.ReadLine();
12        Console.WriteLine("Input the 2nd string : ");
13        str2 = Console.ReadLine();
14        l1 = str1.Length;
15        l2 = str2.Length;
16
17        if (l1 == l2)
18        {
19            for (i = 0; i < l1; i++)
20            {
21                if (str1[i] != str2[i])
22                {
23                    yn = 1;
24                    i = l1;
25                }
26            }
27        }
28
29        if (l1 == l2)
30            flg = 0;
31        else if (l1 > l2)
32            flg = 1;
33        else if (l1 < l2)
34            flg = -1;
35    }
```



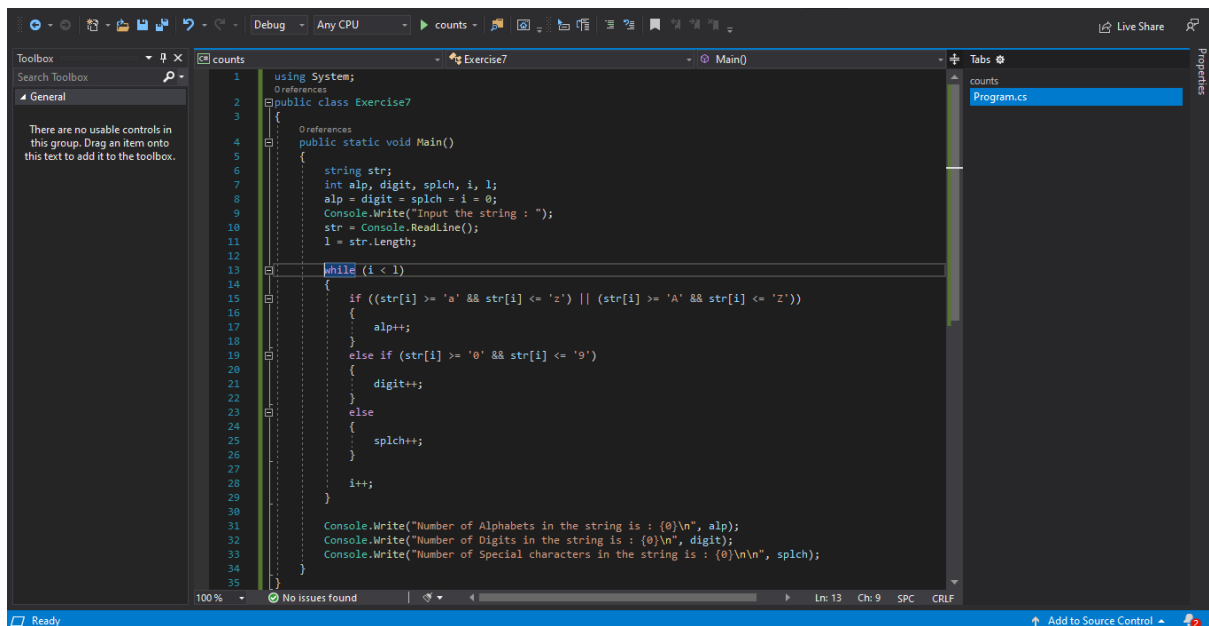
```
36 if (flg == 0)
37 {
38     if (yn == 0)
39         Console.WriteLine("\nThe length of both strings are equal and \nalso, both strings are same.\n\n");
40     else
41         Console.WriteLine("\nThe length of both strings are equal \nbut they are not same.\n\n");
42 }
43 else if (flg == -1)
44 {
45     Console.WriteLine("\nThe length of the first string is smaller than second.\n\n");
46 }
47 else
48 {
49     Console.WriteLine("\nThe length of the first string is greater than second.\n\n");
50 }
51 }
52 }
53 }
```

Output:




```
Microsoft Visual Studio Debug Console
Input the 1st string : sajal
Input the 2nd string : rupali
The length of the first string is smaller than second.
```


14] Write a programme to count a total numbers of alphabets , digits and special characters in a string:



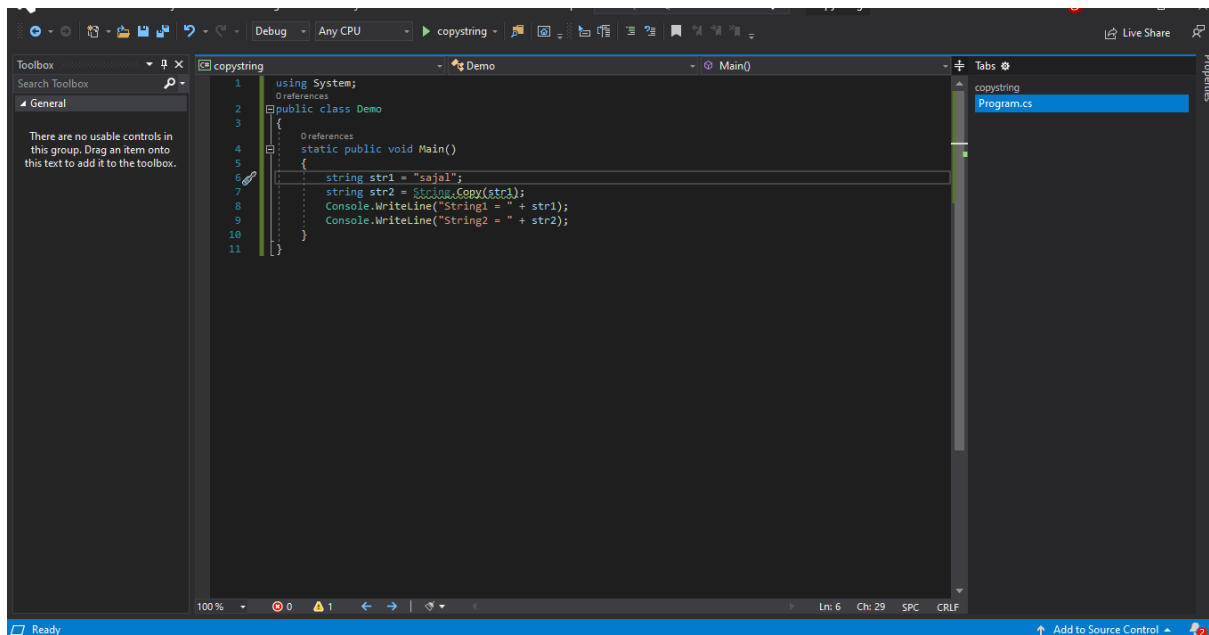
```
1 using System;
2
3 public class Exercise7
4 {
5     public static void Main()
6     {
7         string str;
8         int alp, digit, splch, i, l;
9         alp = digit = splch = i = 0;
10        Console.WriteLine("Input the string : ");
11        str = Console.ReadLine();
12        l = str.Length;
13
14        while (i < l)
15        {
16            if ((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z'))
17            {
18                alp++;
19            }
20            else if (str[i] >= '0' && str[i] <= '9')
21            {
22                digit++;
23            }
24            else
25            {
26                splch++;
27            }
28            i++;
29        }
30
31        Console.WriteLine("Number of Alphabets in the string is : {0}\n", alp);
32        Console.WriteLine("Number of Digits in the string is : {0}\n", digit);
33        Console.WriteLine("Number of Special characters in the string is : {0}\n\n", splch);
34    }
35 }
```

Output:



```
Microsoft Visual Studio Debug Console
Input the string : sajal@123
Number of Alphabets in the string is : 5
Number of Digits in the string is : 3
Number of Special characters in the string is : 1
C:\Users\SANJAY\AppData\Local\Temp\counts\counts\bin\Debug\netcoreapp2.1\counts.exe (process 12504) exited with code 0. To automatically close the console when
```

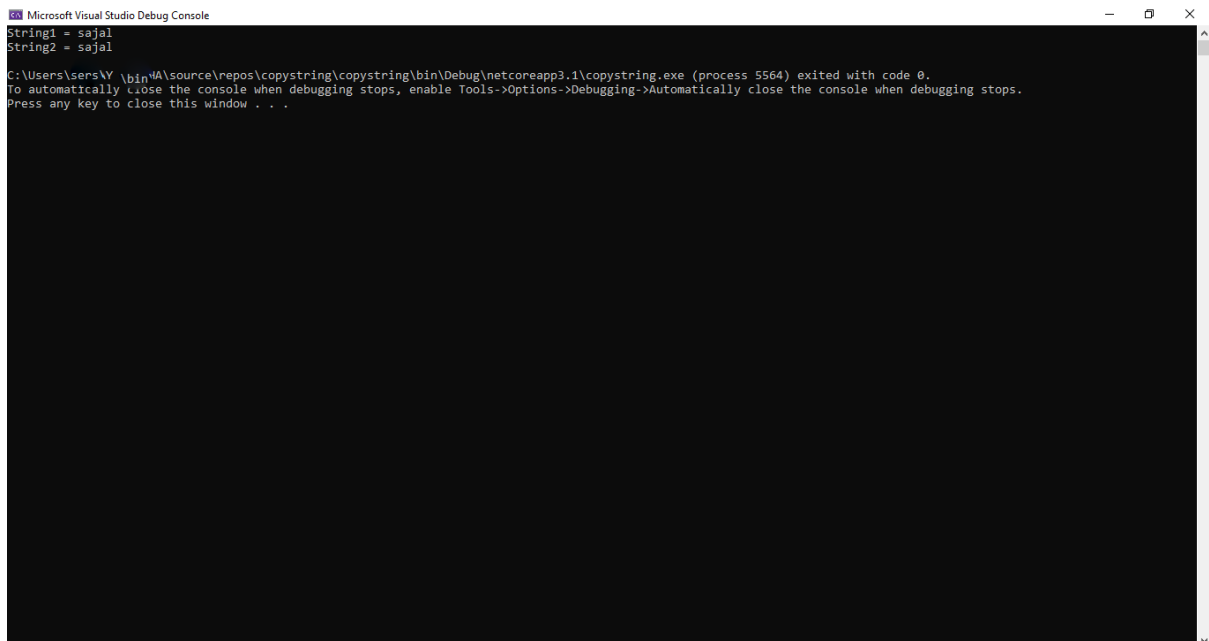
15] Write a programme to copy one string to another string:



The screenshot shows the Visual Studio IDE with a C# project named 'copystring'. The code in 'Program.cs' is as follows:

```
1 using System;
2
3 public class Demo
4 {
5     static public void Main()
6     {
7         string str1 = "sajal";
8         string str2 = String.Copy(str1);
9         Console.WriteLine("String1 = " + str1);
10        Console.WriteLine("String2 = " + str2);
11    }
12 }
```

Output:

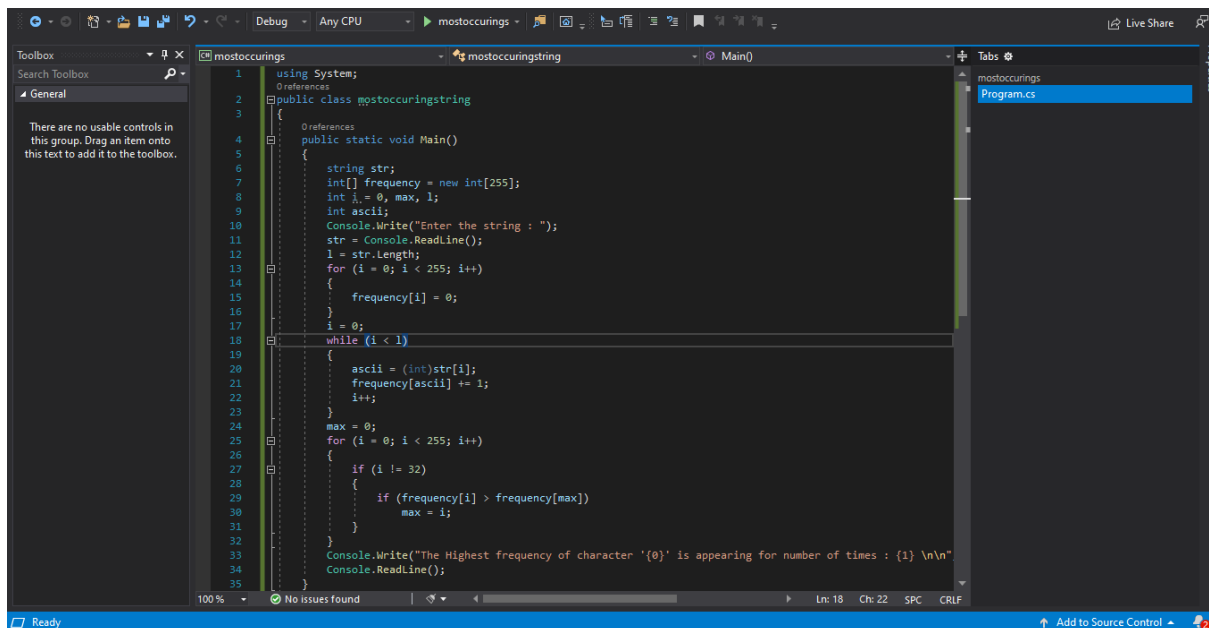


The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
String1 = sajal
String2 = sajal

C:\Users\Users\Y\bin\source\repos\copystring\copystring\bin\Debug\netcoreapp3.1\copystring.exe (process 5564) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

16] Write a programme to find maximum occurring character in a string:



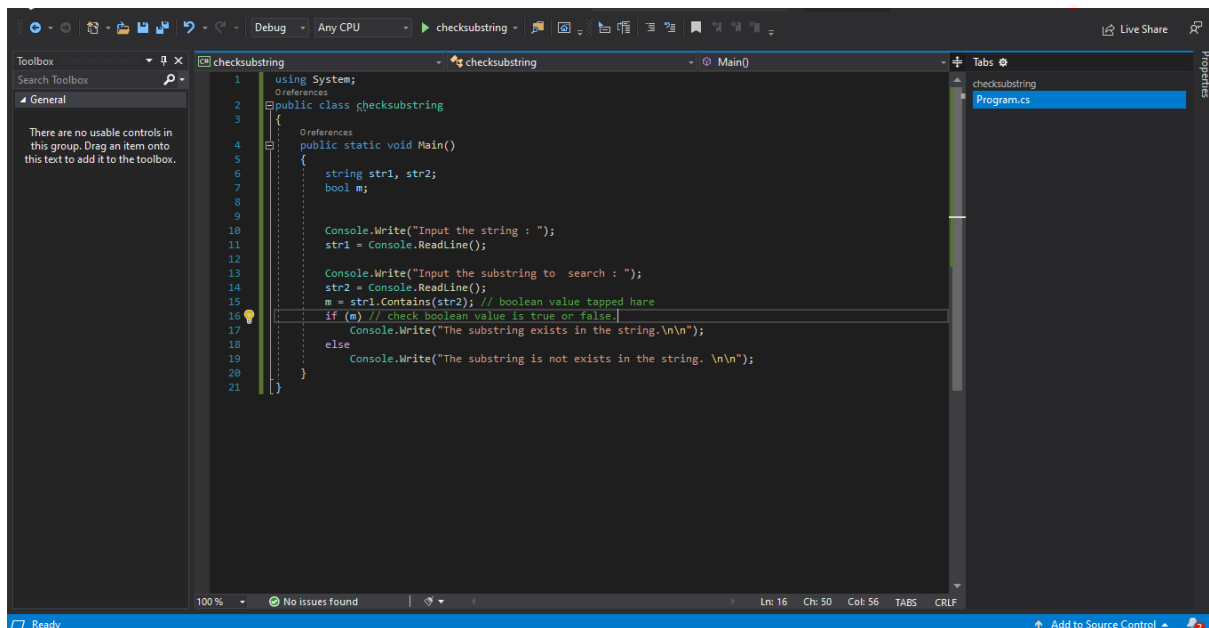
```
1 using System;
2 public class mostoccurringstring
3 {
4     public static void Main()
5     {
6         string str;
7         int[] frequency = new int[255];
8         int i = 0, max = 1;
9         int ascii;
10        Console.WriteLine("Enter the string : ");
11        str = Console.ReadLine();
12        l = str.Length;
13        for (i = 0; i < 255; i++)
14        {
15            frequency[i] = 0;
16        }
17        i = 0;
18        while (i < l)
19        {
20            ascii = (int)str[i];
21            frequency[ascii] += 1;
22            i++;
23        }
24        max = 0;
25        for (i = 0; i < 255; i++)
26        {
27            if (i != 32)
28            {
29                if (frequency[i] > frequency[max])
30                    max = i;
31            }
32        }
33        Console.WriteLine("The Highest frequency of character '{0}' is appearing for number of times : {1} \n\n");
34        Console.ReadLine();
35    }
36 }
```

Output:

C:\Users\SANJAY LADDHA\source\repos\mostoccurringstring\mostoccurringstring\bin\Debug\netcoreapp3.1\mostoccurringstring.exe

```
Enter the string : sajal laddha
The Highest frequency of character 'a' is appearing for number of times : 4
```

17] Write a programme to check whether a given substring is present in the given string:

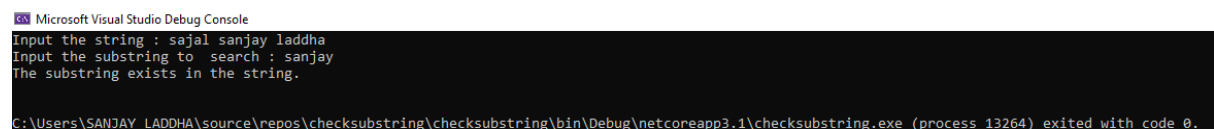


The screenshot shows the Visual Studio IDE with a C# console application named 'checksubstring'. The code is as follows:

```
1 using System;
2 public class checksubstring
3 {
4     public static void Main()
5     {
6         string str1, str2;
7         bool m;
8
9         Console.WriteLine("Input the string : ");
10        str1 = Console.ReadLine();
11
12        Console.WriteLine("Input the substring to search : ");
13        str2 = Console.ReadLine();
14        m = str1.Contains(str2); // boolean value tapped here
15        if (m) // check boolean value is true or false.
16        {
17            Console.WriteLine("The substring exists in the string.\n\n");
18        }
19        else
20        {
21            Console.WriteLine("The substring is not exists in the string. \n\n");
22        }
23    }
24 }
```

Output:

(substring exists)

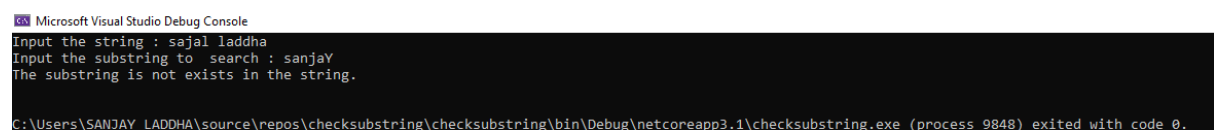


Microsoft Visual Studio Debug Console

```
Input the string : sajal sanjay laddha
Input the substring to search : sanjay
The substring exists in the string.

C:\Users\SANJAY LADDHA\source\repos\checksubstring\checksubstring\bin\Debug\netcoreapp3.1\checksubstring.exe (process 13264) exited with code 0.
```

(substring doesn't exists)

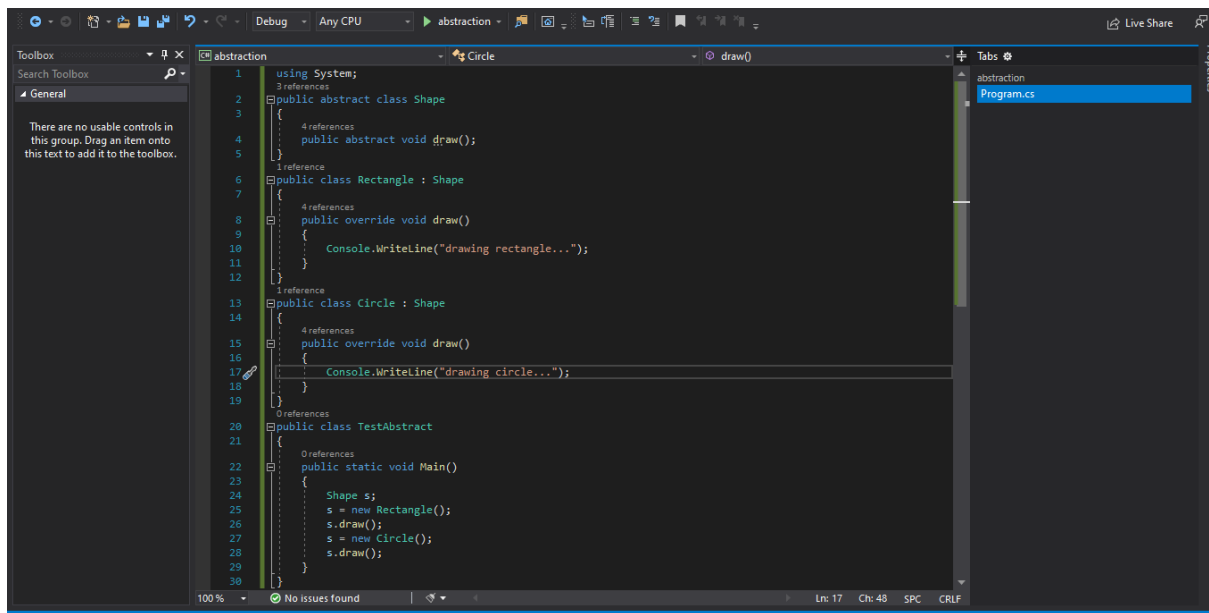


Microsoft Visual Studio Debug Console

```
Input the string : sajal laddha
Input the substring to search : sanjaY
The substring is not exists in the string.

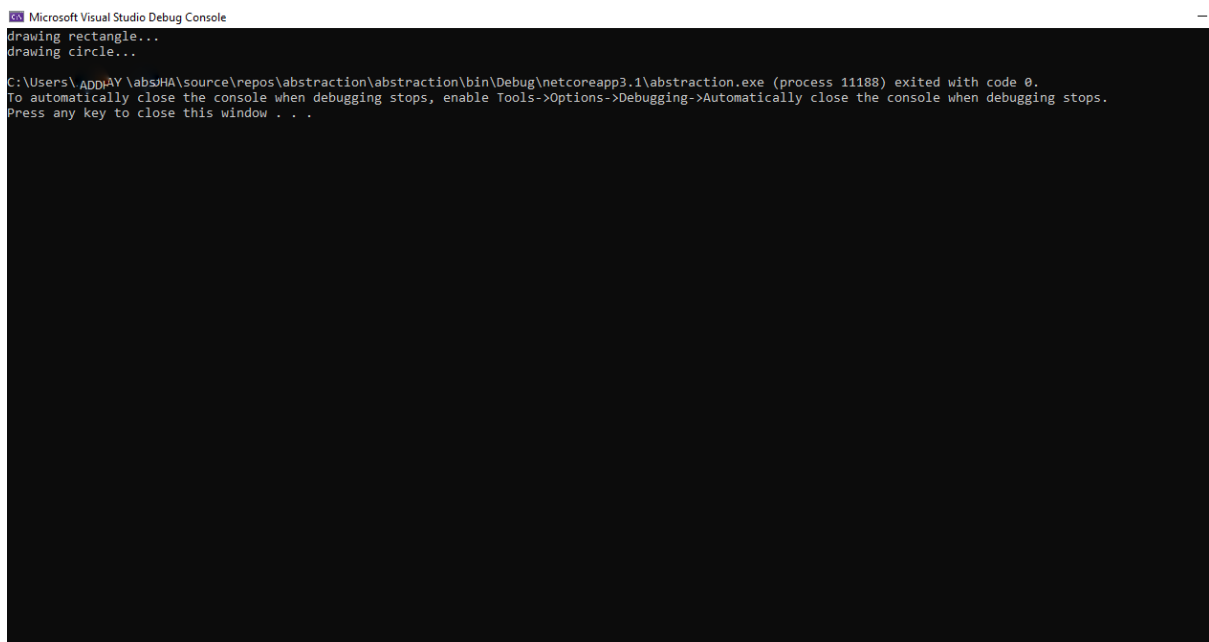
C:\Users\SANJAY LADDHA\source\repos\checksubstring\checksubstring\bin\Debug\netcoreapp3.1\checksubstring.exe (process 9848) exited with code 0.
```

18] Write a programme for abstraction:



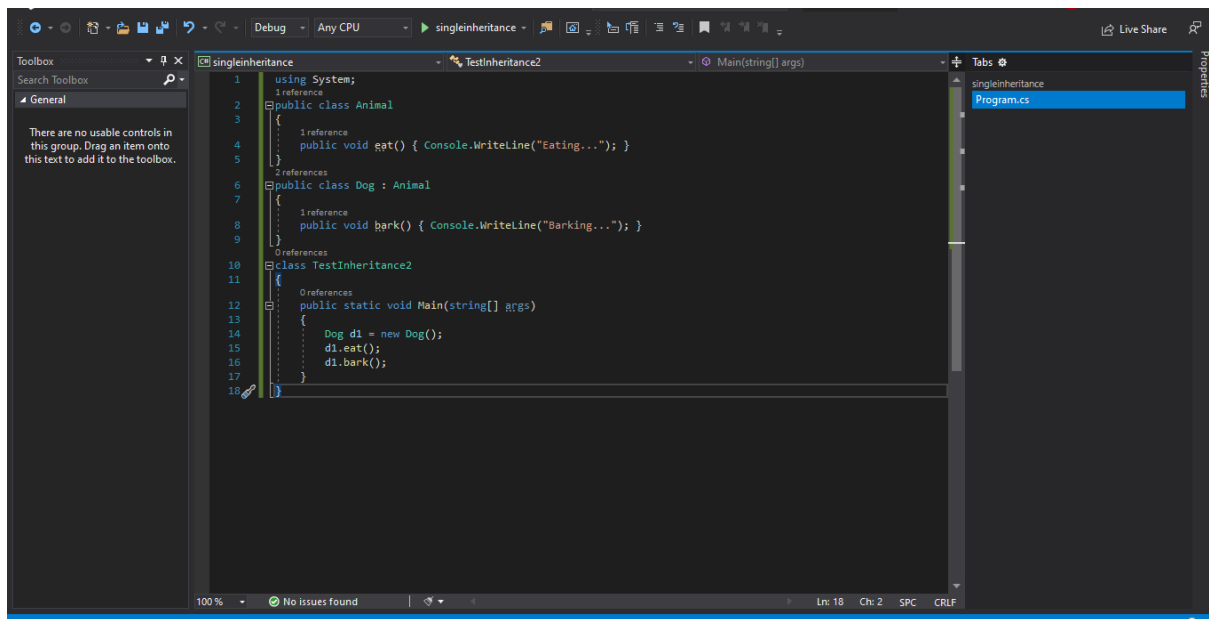
```
1 using System;
2 public abstract class Shape
3 {
4     public abstract void draw();
5 }
6 public class Rectangle : Shape
7 {
8     public override void draw()
9     {
10         Console.WriteLine("drawing rectangle...");
11     }
12 }
13 public class Circle : Shape
14 {
15     public override void draw()
16     {
17         Console.WriteLine("drawing circle...");
18     }
19 }
20 public class TestAbstract
21 {
22     public static void Main()
23     {
24         Shape s;
25         s = new Rectangle();
26         s.draw();
27         s = new Circle();
28         s.draw();
29     }
30 }
```

Output:



```
Microsoft Visual Studio Debug Console
drawing rectangle...
drawing circle...
C:\Users\appu\AY\labs\HA\source\repos\abstraction\abstraction\bin\Debug\netcoreapp3.1\abstraction.exe (process 11188) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

19] Write a programme for single inheritance:



The screenshot shows the Visual Studio IDE with a C# project named 'singleinheritance'. The code is as follows:

```
1 using System;
2 public class Animal
3 {
4     public void eat() { Console.WriteLine("Eating..."); }
5 }
6 public class Dog : Animal
7 {
8     public void bark() { Console.WriteLine("Barking..."); }
9 }
10 class TestInheritance2
11 {
12     public static void Main(string[] args)
13     {
14         Dog d1 = new Dog();
15         d1.eat();
16         d1.bark();
17     }
18 }
```

The interface includes a Toolbox on the left, a Solution Explorer on the right showing 'Program.cs', and a status bar at the bottom indicating 'No issues found'.

Output:

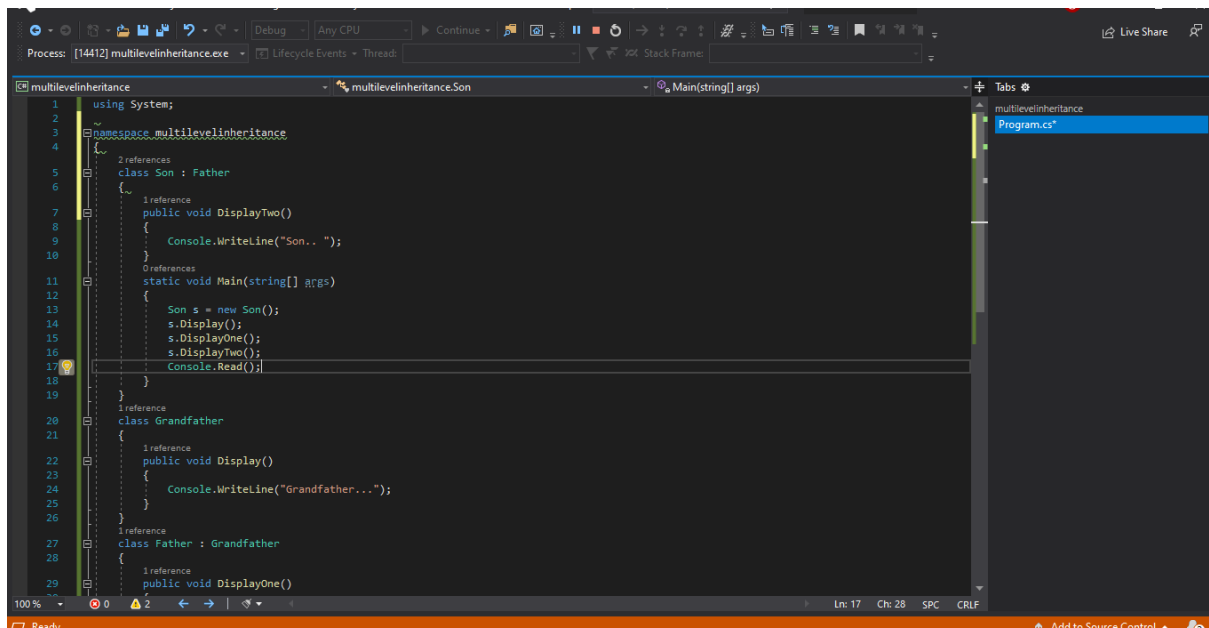


The screenshot shows the 'Microsoft Visual Studio Debug Console' window. The output of the program is:

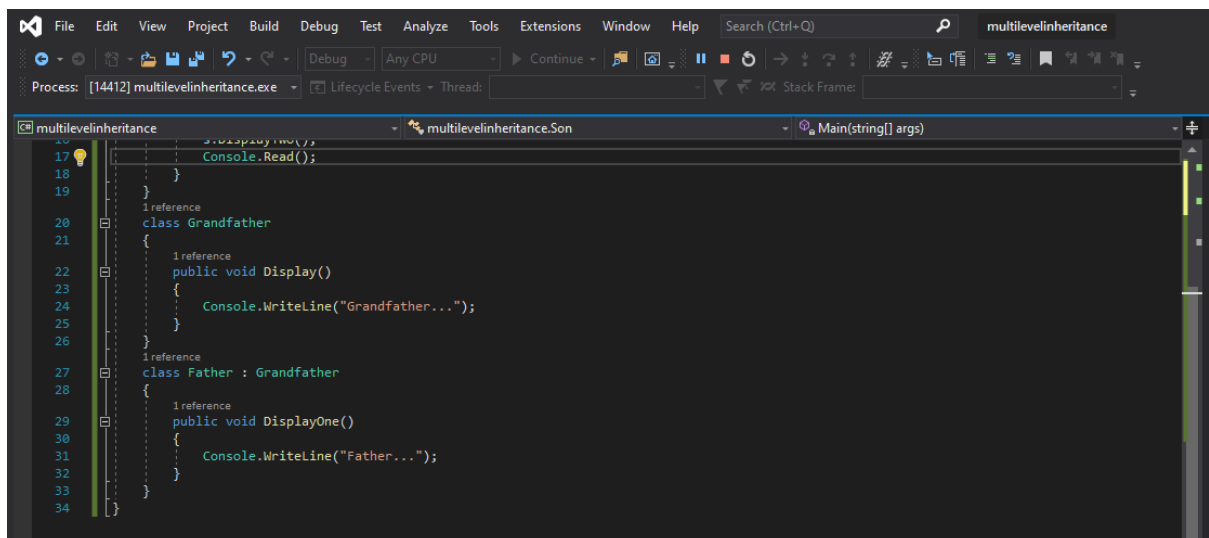
```
Eating...
Barking...

C:\Users\96171\AppData\source\repos\singleinheritance\singleinheritance\bin\Debug\netcoreapp3.1\singleinheritance.exe (process 12256) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

20] Write a programme for multi level inheritance:



```
1 using System;
2
3 namespace multilevelinheritance
4 {
5     2 references
6     class Son : Father
7     {
8         1 reference
9         public void DisplayTwo()
10        {
11            Console.WriteLine("Son.. ");
12        }
13    }
14    0 references
15    static void Main(string[] args)
16    {
17        Son s = new Son();
18        s.Display();
19        s.DisplayOne();
20        s.DisplayTwo();
21        Console.ReadLine();
22    }
23 }
24 1 reference
25 class Grandfather
26 {
27     1 reference
28     public void Display()
29     {
30         Console.WriteLine("Grandfather...");
31     }
32 }
33 1 reference
34 class Father : Grandfather
35 {
36     1 reference
37     public void DisplayOne()
38     {
39         Console.WriteLine("Father...");
40     }
41 }
```



```
17 Console.ReadLine();
18 }
19 }
20 1 reference
21 class Grandfather
22 {
23     1 reference
24     public void Display()
25     {
26         Console.WriteLine("Grandfather...");
27     }
28 }
29 1 reference
30 class Father : Grandfather
31 {
32     1 reference
33     public void DisplayOne()
34     {
35         Console.WriteLine("Father...");
36     }
37 }
```

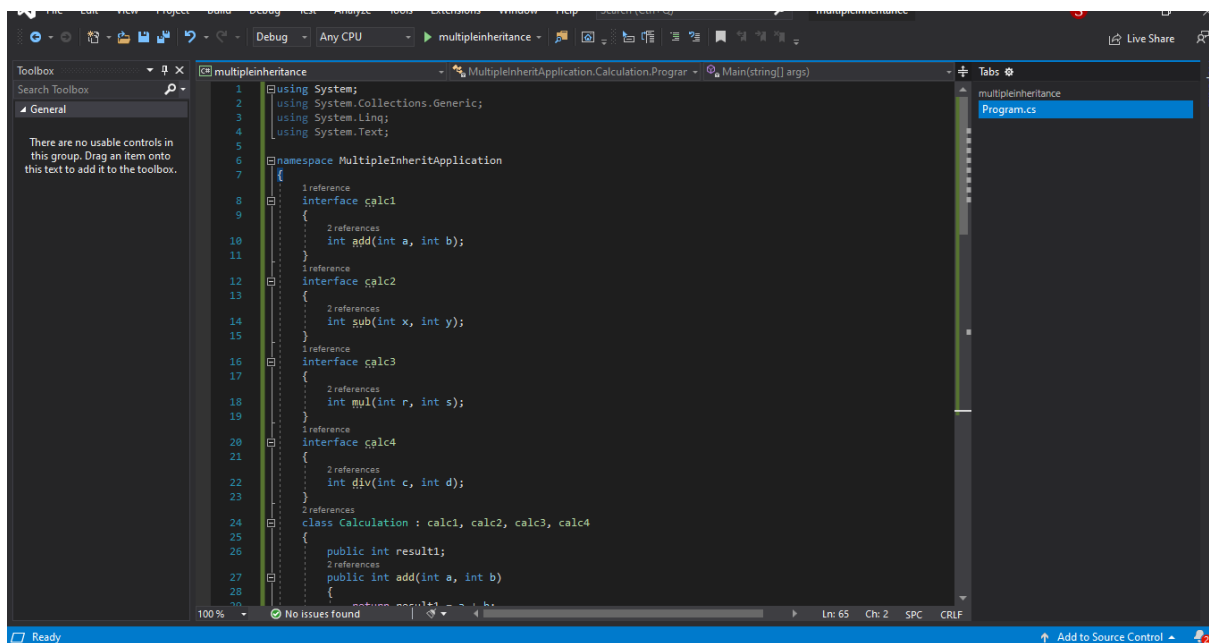
Output:



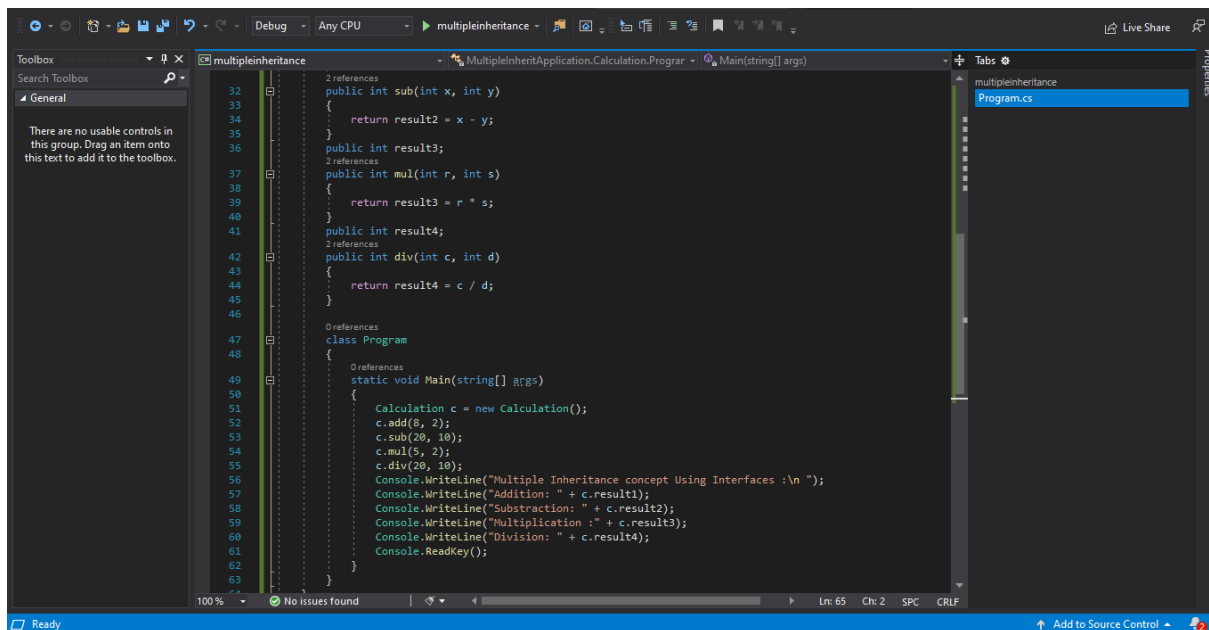
```
Grandfather...
Father...
Son..
```

21] Write a programme for multiple inheritance:

(Multiple inheritance runs with the help of interface not class)

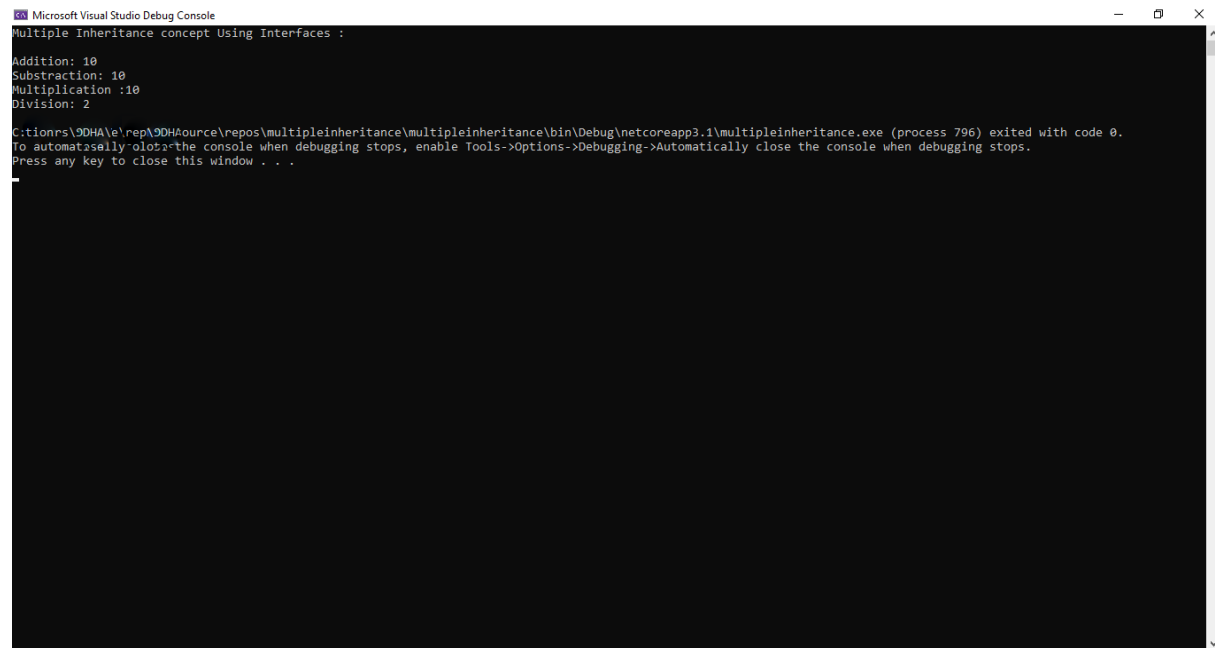


```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5
6 namespace MultipleInheritApplication
7 {
8     1 reference
9     interface calc1
10     {
11         2 references
12         int add(int a, int b);
13     }
14     1 reference
15     interface calc2
16     {
17         2 references
18         int sub(int x, int y);
19     }
20     1 reference
21     interface calc3
22     {
23         2 references
24         int mul(int r, int s);
25     }
26     1 reference
27     interface calc4
28     {
29         2 references
30         int div(int c, int d);
31     }
32     2 references
33     class Calculation : calc1, calc2, calc3, calc4
34     {
35         public int result1;
36         2 references
37         public int add(int a, int b)
38         {
39             return result1 = a + b;
40         }
41     }
42 }
```



```
32     2 references
33     public int sub(int x, int y)
34     {
35         return result2 = x - y;
36     }
37     public int result3;
38     2 references
39     public int mul(int r, int s)
40     {
41         return result3 = r * s;
42     }
43     public int result4;
44     2 references
45     public int div(int c, int d)
46     {
47         return result4 = c / d;
48     }
49
50     0 references
51     class Program
52     {
53         0 references
54         static void Main(string[] Args)
55         {
56             Calculation c = new Calculation();
57             c.add(8, 2);
58             c.sub(20, 10);
59             c.mul(5, 2);
60             c.div(20, 10);
61             Console.WriteLine("Multiple Inheritance concept Using Interfaces :\n ");
62             Console.WriteLine("Addition: " + c.result1);
63             Console.WriteLine("Substraction: " + c.result2);
64             Console.WriteLine("Multiplication: " + c.result3);
65             Console.WriteLine("Division: " + c.result4);
66             Console.ReadKey();
67         }
68     }
69 }
```

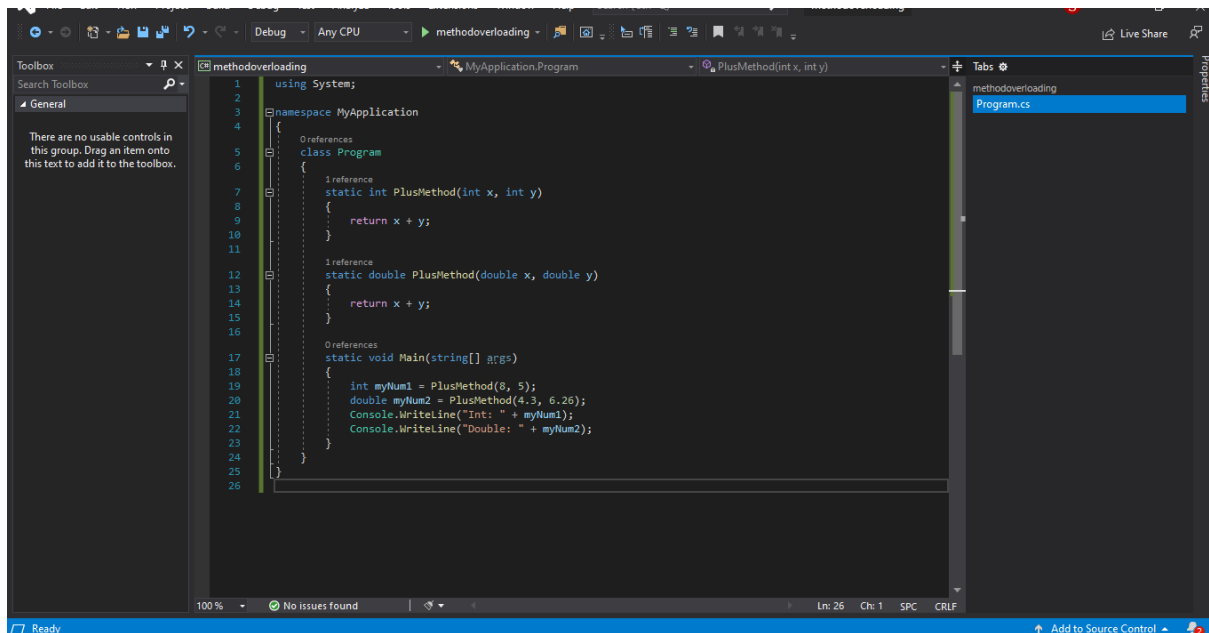

Output:



```
Microsoft Visual Studio Debug Console
Multiple Inheritance concept Using Interfaces :
Addition: 10
Substraction: 10
Multiplication :10
Division: 2

C:\Users\90DHA\source\repos\multipleinheritance\multipleinheritance\bin\Debug\netcoreapp3.1\multipleinheritance.exe (process 796) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

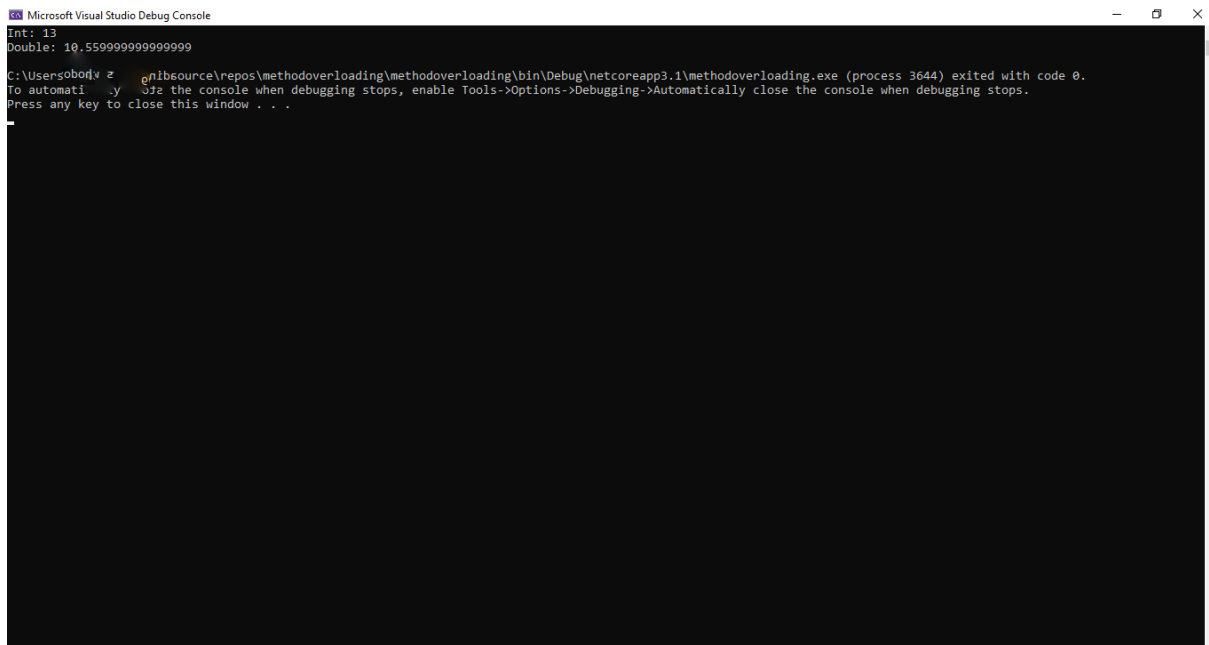
22] Write a programme for method overloading:



The screenshot shows the Visual Studio IDE with a C# project named 'methodoverloading'. The code is as follows:

```
1 using System;
2
3 namespace MyApplication
4 {
5     References
6     class Program
7     {
8         static int PlusMethod(int x, int y)
9         {
10             return x + y;
11         }
12
13         static double PlusMethod(double x, double y)
14         {
15             return x + y;
16         }
17
18         static void Main(string[] args)
19         {
20             int myNum1 = PlusMethod(8, 5);
21             double myNum2 = PlusMethod(4.3, 6.26);
22             Console.WriteLine("Int: " + myNum1);
23             Console.WriteLine("Double: " + myNum2);
24         }
25     }
26 }
```

Output:

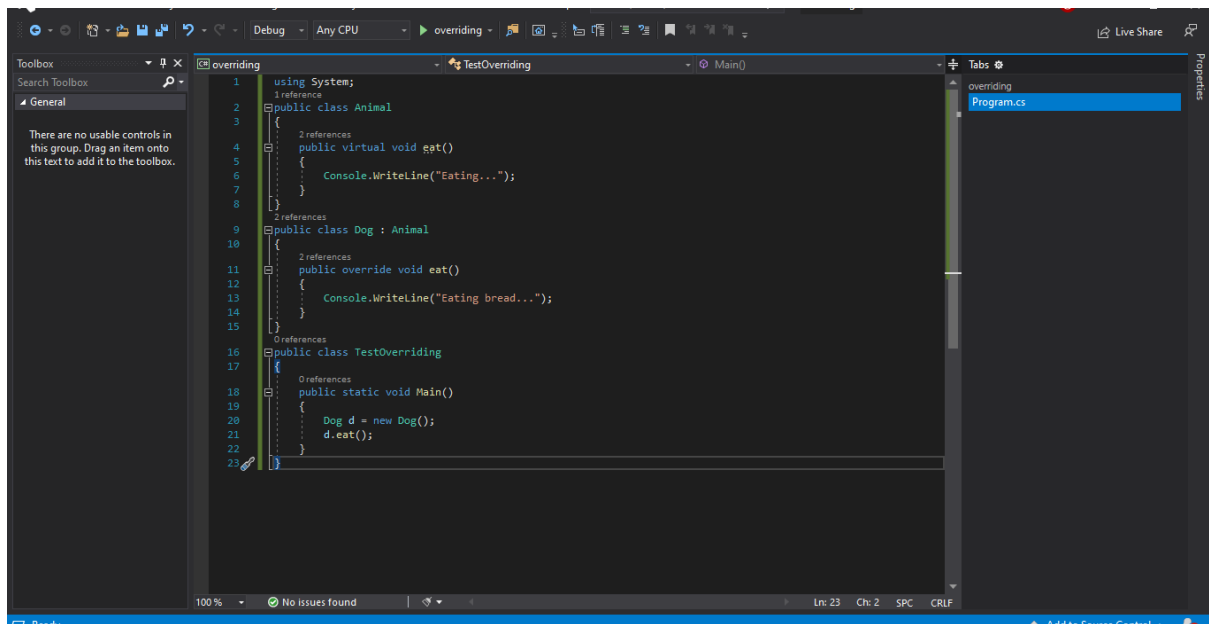


The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
Int: 13
Double: 10.559999999999999

C:\Users\boris\z\@nibsource\repos\methodoverloading\methodoverloading\bin\Debug\netcoreapp3.1\methodoverloading.exe (process 3644) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

23] Write a programme for method overriding:

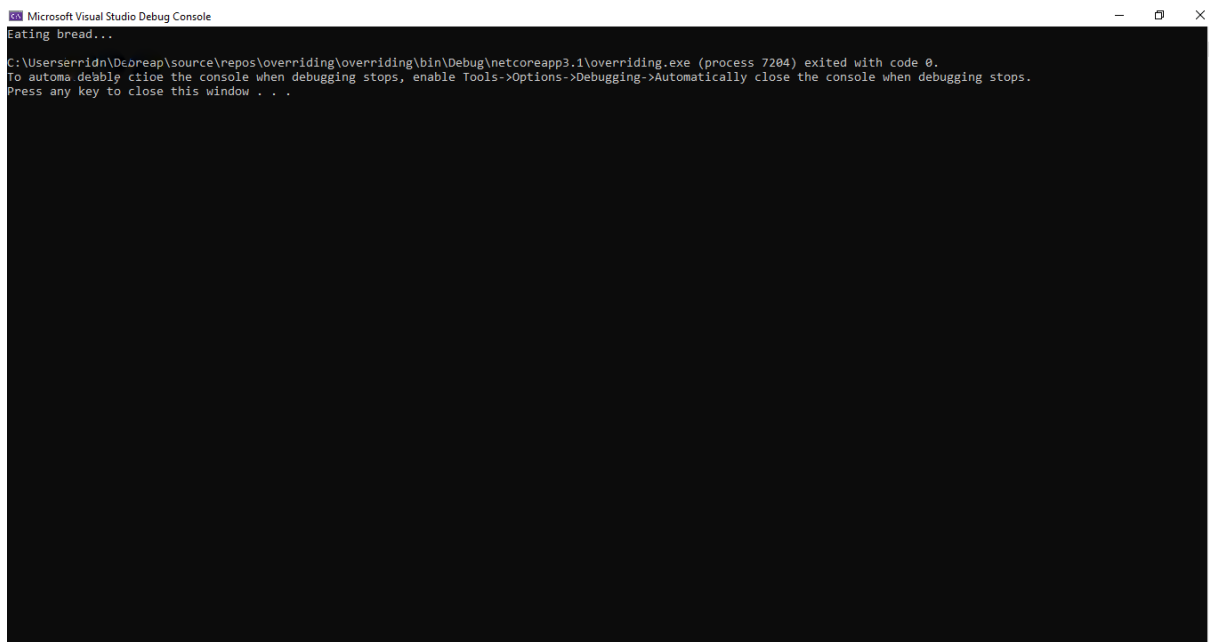


The screenshot shows the Visual Studio IDE with a C# project named 'overriding'. The code is as follows:

```
1 using System;
2 public class Animal
3 {
4     public virtual void eat()
5     {
6         Console.WriteLine("Eating...");
7     }
8 }
9
10 public class Dog : Animal
11 {
12     public override void eat()
13     {
14         Console.WriteLine("Eating bread...");
15     }
16 }
17
18 public class TestOverriding
19 {
20     public static void Main()
21     {
22         Dog d = new Dog();
23         d.eat();
24     }
25 }
```

The code demonstrates method overriding where the 'Dog' class overrides the 'eat()' method of the 'Animal' class. The 'TestOverriding' class creates an instance of 'Dog' and calls the 'eat()' method, which outputs 'Eating bread...'.

Output:

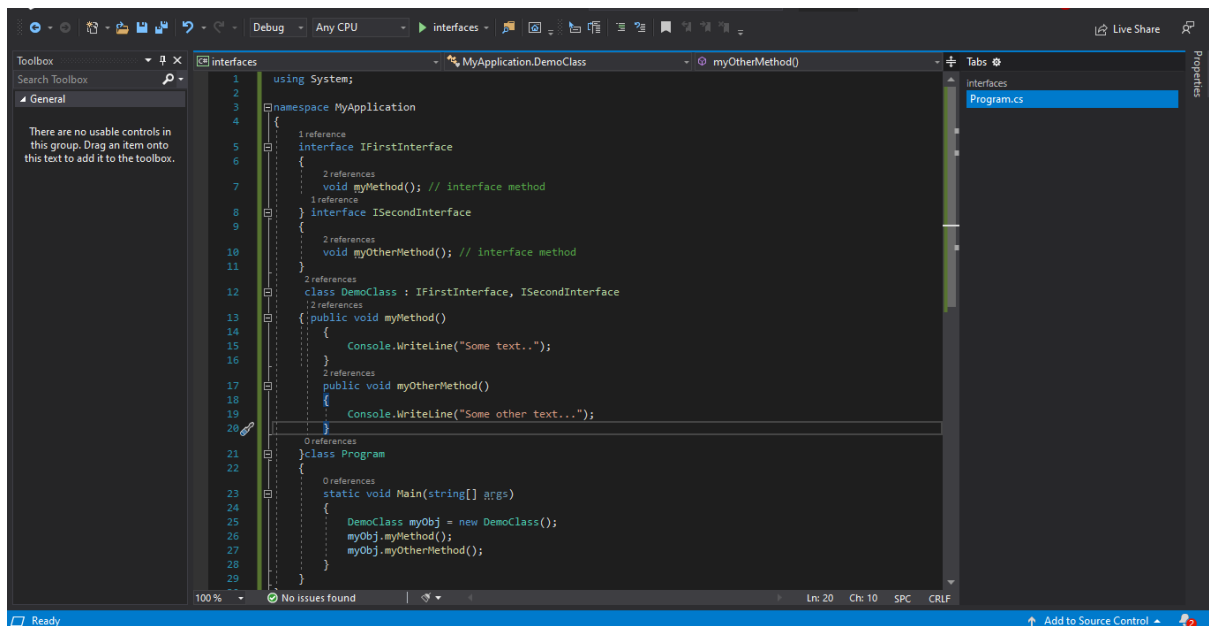


The screenshot shows the Microsoft Visual Studio Debug Console. The output is as follows:

```
Eating bread...
C:\Users\srnidn\Documents\source\repos\overriding\overriding\bin\Debug\netcoreapp3.1\overriding.exe (process 7204) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

The output shows the program running successfully and exiting with code 0. The console window is waiting for a key press to close.

24] Write a programme for interface:

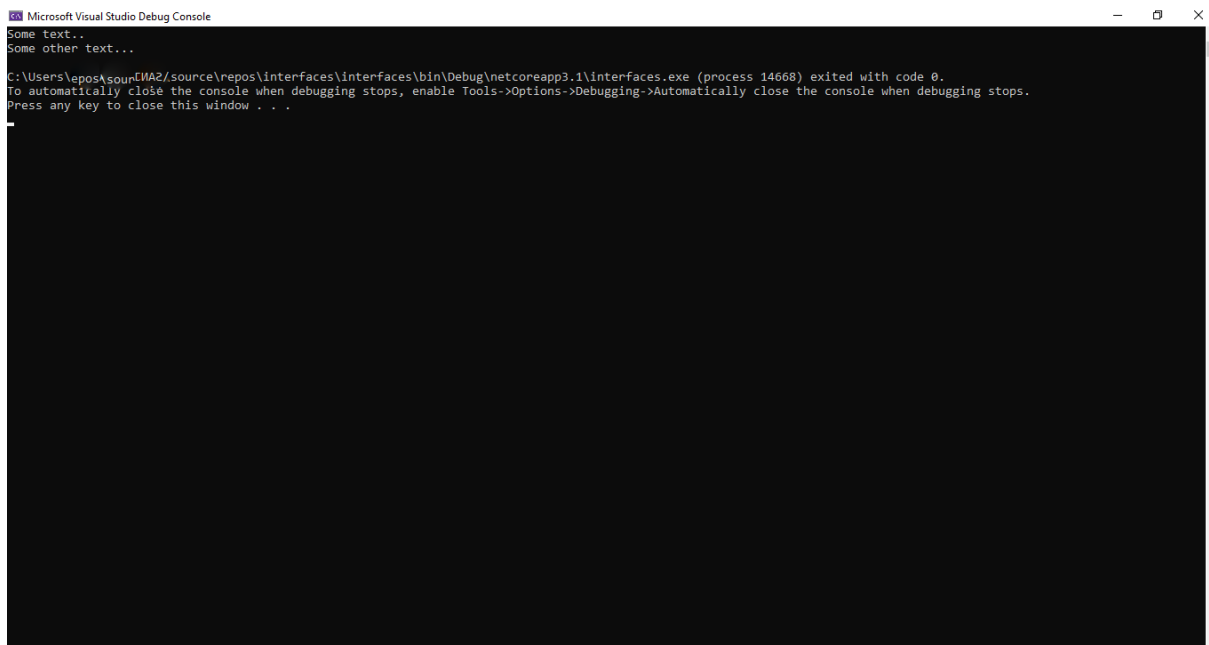


The screenshot shows the Visual Studio IDE with a C# project named 'interfaces'. The main editor displays the following code:

```
1 using System;
2
3 namespace MyApplication
4 {
5     1 reference
6     interface IFirstInterface
7     {
8         2 references
9         void myMethod(); // interface method
10    }
11    1 reference
12    interface ISecondInterface
13    {
14        2 references
15        void myOtherMethod(); // interface method
16    }
17    2 references
18    class DemoClass : IFirstInterface, ISecondInterface
19    {
20        2 references
21        public void myMethod()
22        {
23            Console.WriteLine("Some text..");
24        }
25        2 references
26        public void myOtherMethod()
27        {
28            Console.WriteLine("Some other text...");
29        }
30    }
31 }
32
33 0 references
34 class Program
35 {
36    0 references
37    static void Main(string[] args)
38    {
39        DemoClass myObj = new DemoClass();
40        myObj.myMethod();
41        myObj.myOtherMethod();
42    }
43 }
```

The interface 'IFirstInterface' has a method 'myMethod()' and 'ISecondInterface' has a method 'myOtherMethod()'. The 'DemoClass' implements both interfaces. The 'Program' class has a 'Main' method that creates a 'DemoClass' object and calls both methods. The status bar at the bottom indicates '100%' zoom, 'No issues found', and 'Ln: 20 Ch: 10 SPC CRLF'.

Output:



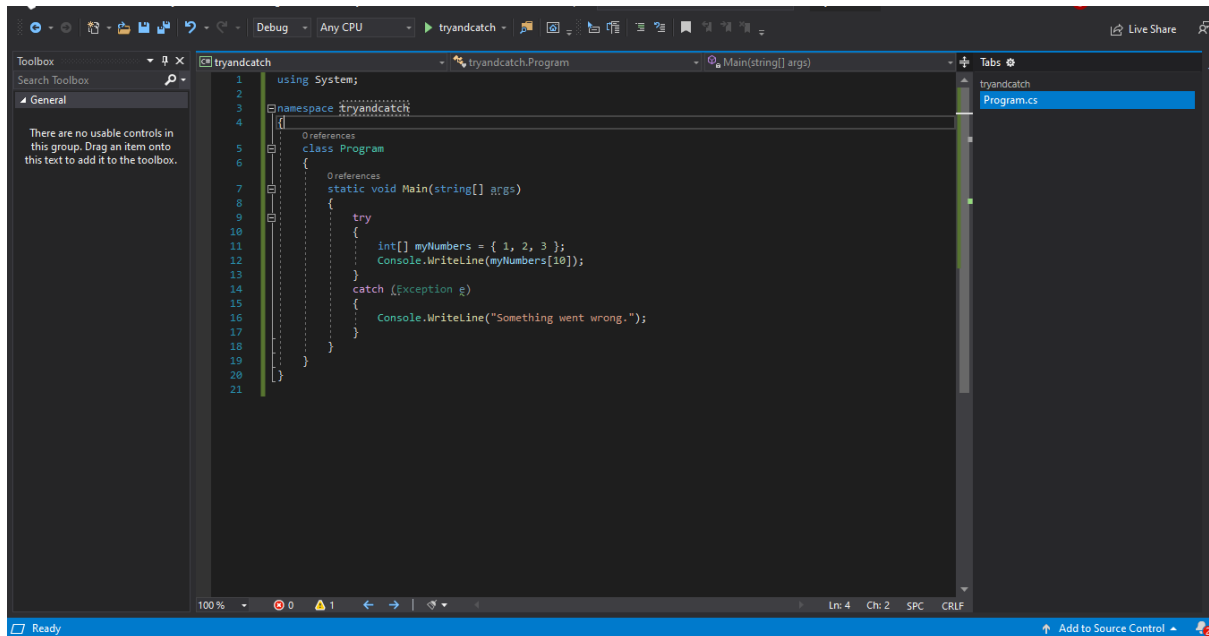
The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
Some text..
Some other text...
```

Below the output, the console shows the command prompt path and the exit code:

```
C:\Users\reposit\source\repos\interfaces\interfaces\bin\Debug\netcoreapp3.1\interfaces.exe (process 14668) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

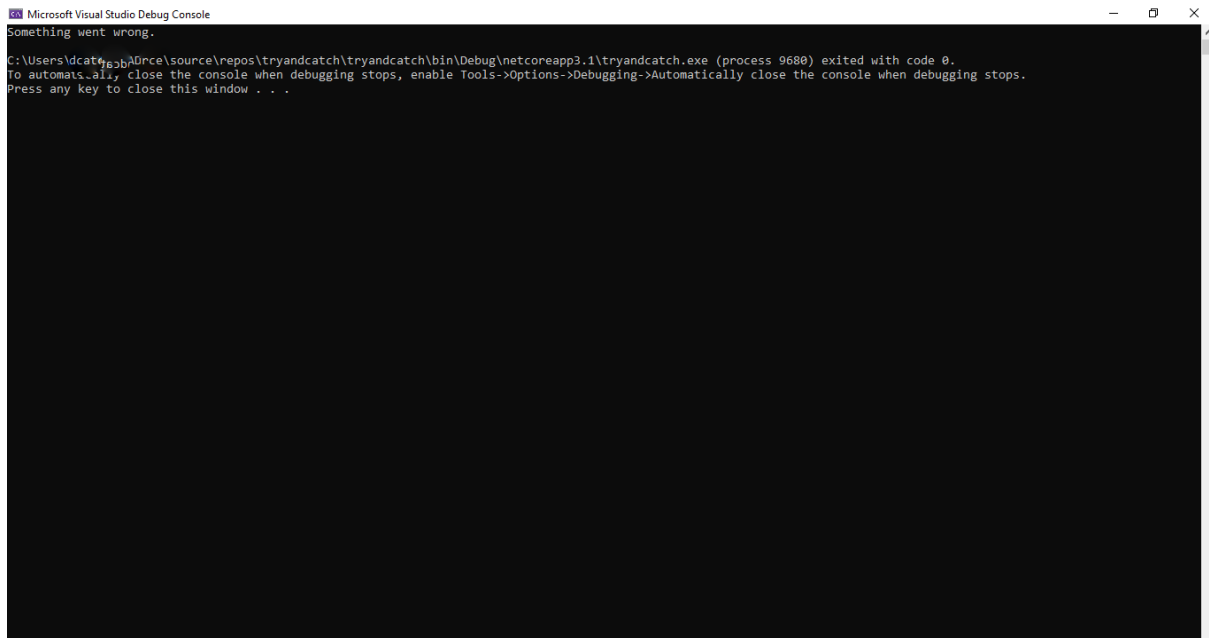
25] Write a programme for exception handling through try and catch:



The screenshot shows the Visual Studio IDE with a C# project named 'tryandcatch'. The code is as follows:

```
1 using System;
2
3 namespace tryandcatch
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             try
10             {
11                 int[] myNumbers = { 1, 2, 3 };
12                 Console.WriteLine(myNumbers[10]);
13             }
14             catch (Exception e)
15             {
16                 Console.WriteLine("Something went wrong.");
17             }
18         }
19     }
20 }
21
```

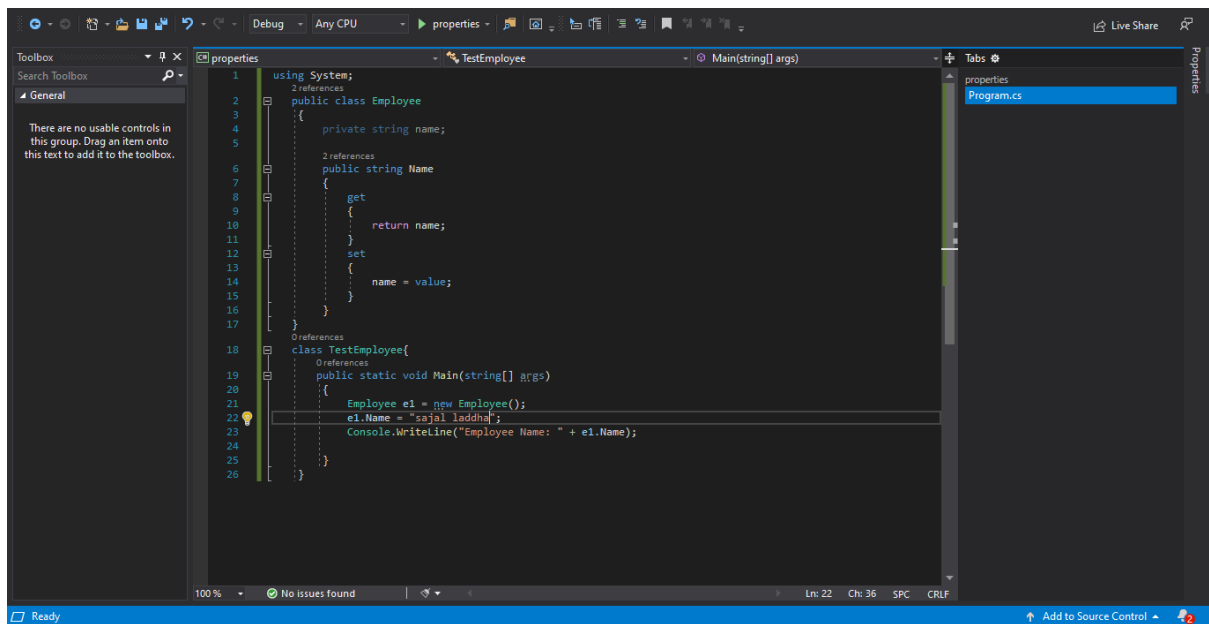
Output:



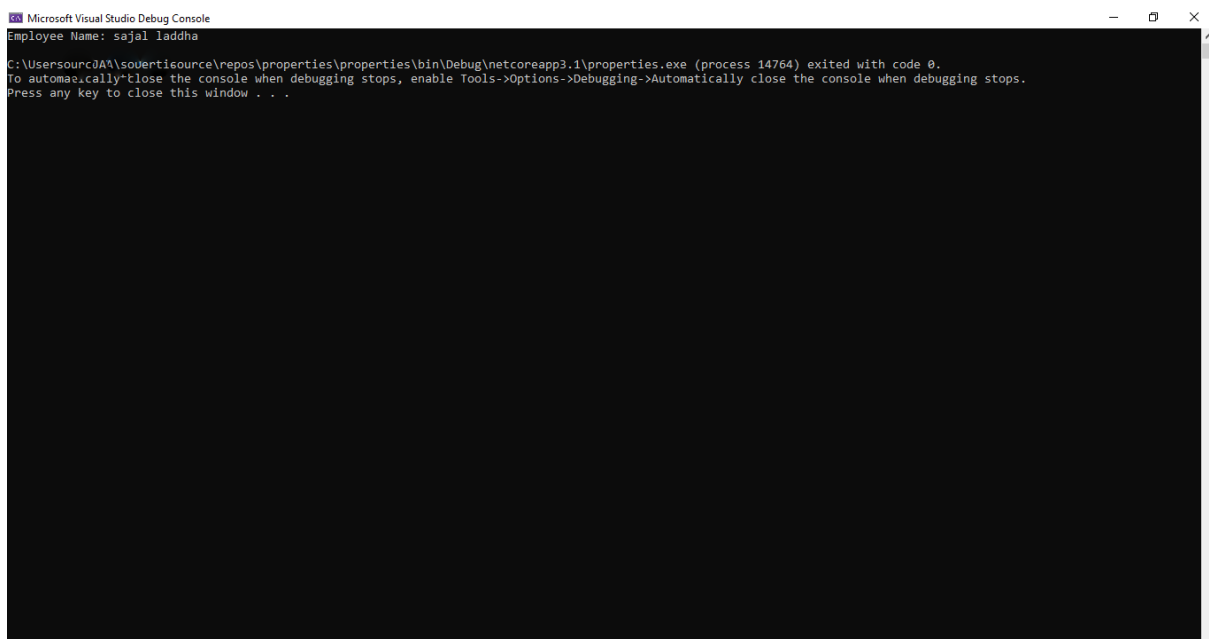
The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
Microsoft Visual Studio Debug Console
Something went wrong.
C:\Users\idcaty65b\OneDrive\source\repos\tryandcatch\tryandcatch\bin\Debug\netcoreapp3.1\tryandcatch.exe (process 9680) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

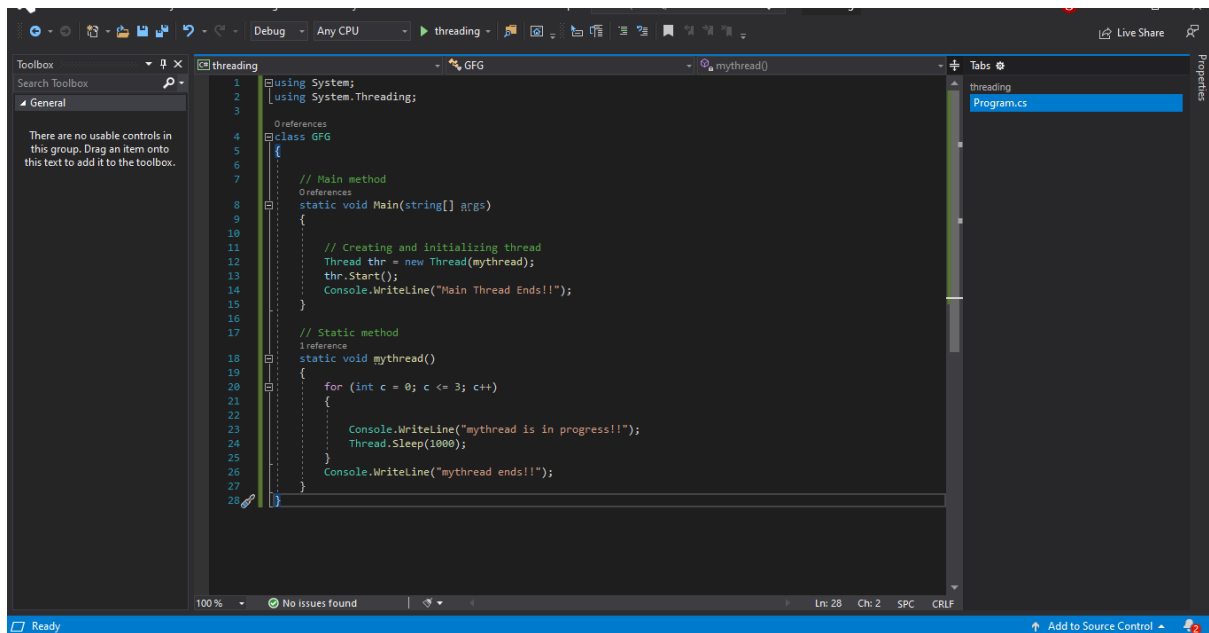
26] Write a programme for properties.



Output:



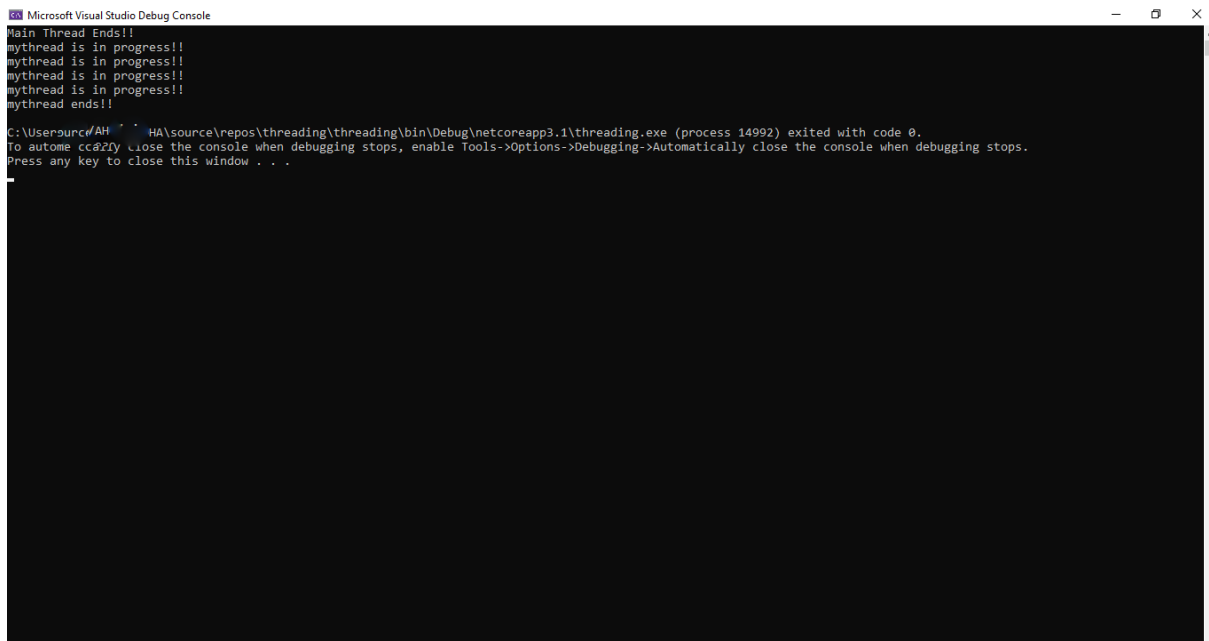
27] Write a programme for threading.



The screenshot shows the Visual Studio IDE with a C# project named 'threading'. The code is as follows:

```
1 using System;
2 using System.Threading;
3
4 class GFG
5 {
6
7     // Main method
8     static void Main(string[] args)
9     {
10
11         // Creating and initializing thread
12         Thread thr = new Thread(mythread);
13         thr.Start();
14         Console.WriteLine("Main Thread Ends!!");
15     }
16
17     // Static method
18     static void mythread()
19     {
20         for (int c = 0; c <= 3; c++)
21         {
22             Console.WriteLine("mythread is in progress!!");
23             Thread.Sleep(1000);
24         }
25         Console.WriteLine("mythread ends!!");
26     }
27
28 }
```

Output:

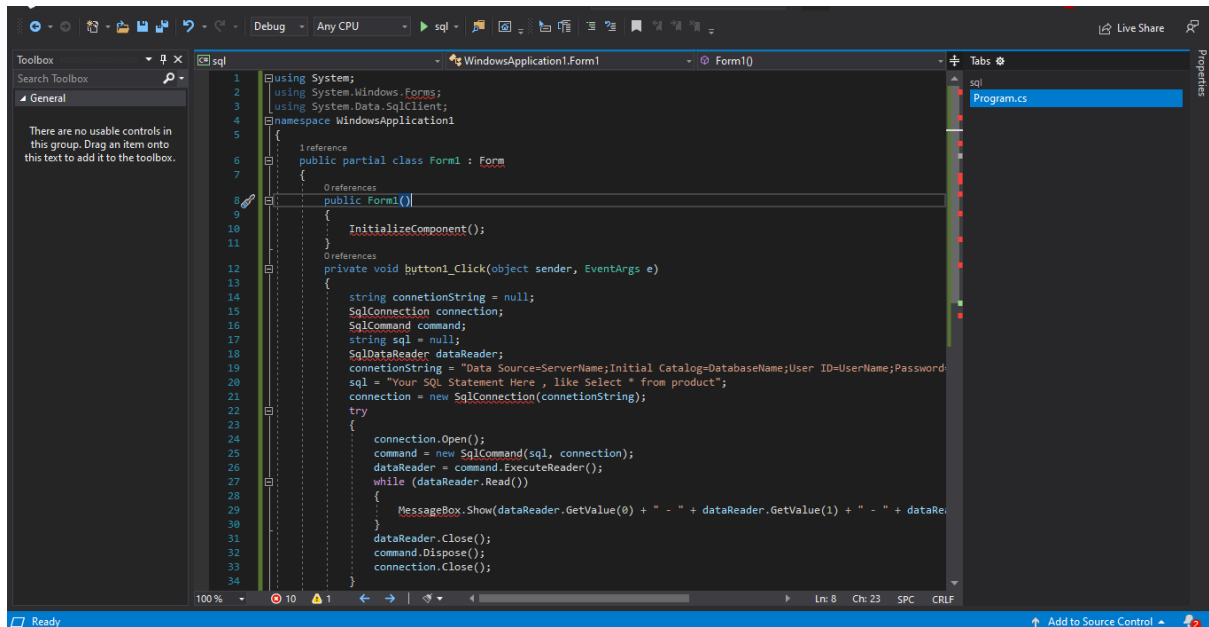


The screenshot shows the Microsoft Visual Studio Debug Console with the following output:

```
Main Thread Ends!!
mythread is in progress!!
mythread is in progress!!
mythread is in progress!!
mythread is in progress!!
mythread ends!!

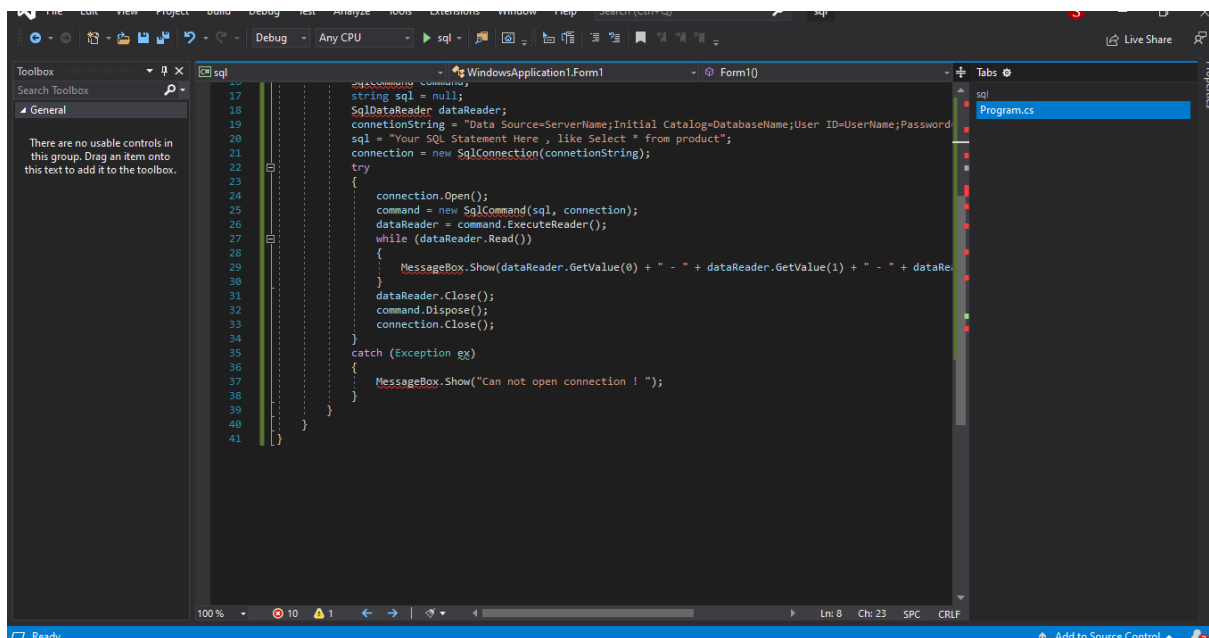
C:\Users\rc\AH\source\repos\threading\threading\bin\Debug\netcoreapp3.1\threading.exe (process 14992) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

28] Write a programme to access data from database using ADO.net.



This screenshot shows the first part of a C# program in Visual Studio. The code is for a Windows Form application. It includes the following code:

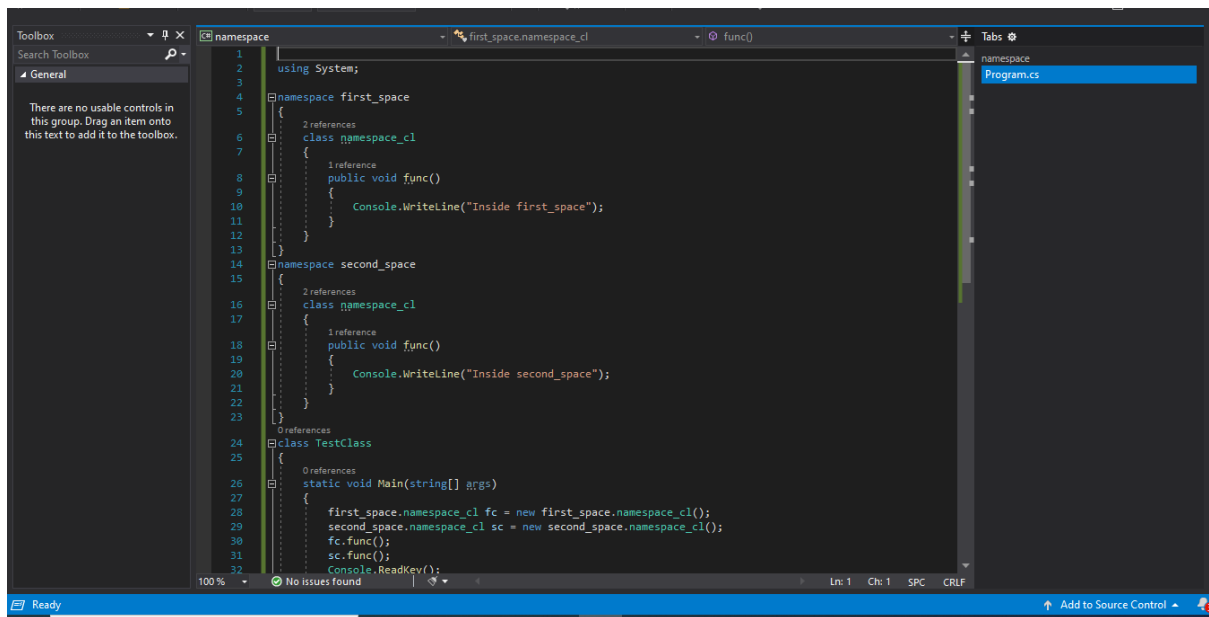
```
1 using System;
2 using System.Windows.Forms;
3 using System.Data.SqlClient;
4 namespace WindowsApplication1
5 {
6     [reference]
7     public partial class Form1 : Form
8     {
9         [References]
10        public Form1()
11        {
12            InitializeComponent();
13        }
14        [References]
15        private void button1_Click(object sender, EventArgs e)
16        {
17            string connetionString = null;
18            SqlConnection connection;
19            SqlCommand command;
20            string sql = null;
21            SqlDataReader dataReader;
22            connetionString = "Data Source=ServerName;Initial Catalog=DatabaseName;User ID=UserName;Password=Password";
23            sql = "Your SQL Statement Here , like Select * from product";
24            connection = new SqlConnection(connetionString);
25            try
26            {
27                connection.Open();
28                command = new SqlCommand(sql, connection);
29                dataReader = command.ExecuteReader();
30                while (dataReader.Read())
31                {
32                    MessageBox.Show(dataReader.GetValue(0) + " - " + dataReader.GetValue(1) + " - " + dataReader.GetValue(2));
33                }
34                dataReader.Close();
35                command.Dispose();
36                connection.Close();
37            }
38            catch (Exception ex)
39            {
40                MessageBox.Show("Can not open connection ! ");
41            }
42        }
43    }
44 }
```



This screenshot shows the second part of the C# program in Visual Studio. The code is for a Windows Form application. It includes the following code:

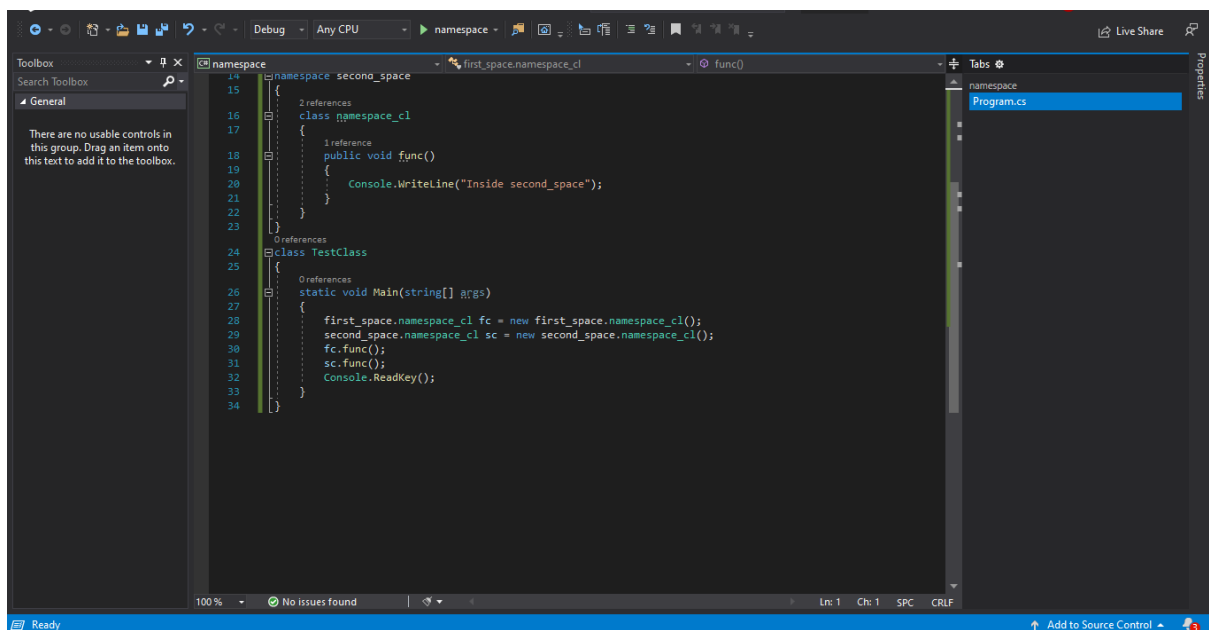
```
17 string sql = null;
18 SqlDataReader dataReader;
19 connetionString = "Data Source=ServerName;Initial Catalog=DatabaseName;User ID=UserName;Password=Password";
20 sql = "Your SQL Statement Here , like Select * from product";
21 connection = new SqlConnection(connetionString);
22 try
23 {
24     connection.Open();
25     command = new SqlCommand(sql, connection);
26     dataReader = command.ExecuteReader();
27     while (dataReader.Read())
28     {
29         MessageBox.Show(dataReader.GetValue(0) + " - " + dataReader.GetValue(1) + " - " + dataReader.GetValue(2));
30     }
31     dataReader.Close();
32     command.Dispose();
33     connection.Close();
34 }
35 catch (Exception ex)
36 {
37     MessageBox.Show("Can not open connection ! ");
38 }
39 }
40 }
41 }
```


29] Write a programme using namespace.



This screenshot shows the initial state of a C# program in Visual Studio. The code defines two namespaces, `first_space` and `second_space`, each containing a `namespace_cl` class with a `func()` method. A `TestClass` contains a `Main` method that instantiates objects from both namespaces and calls their `func()` methods. The status bar at the bottom indicates 'No issues found'.

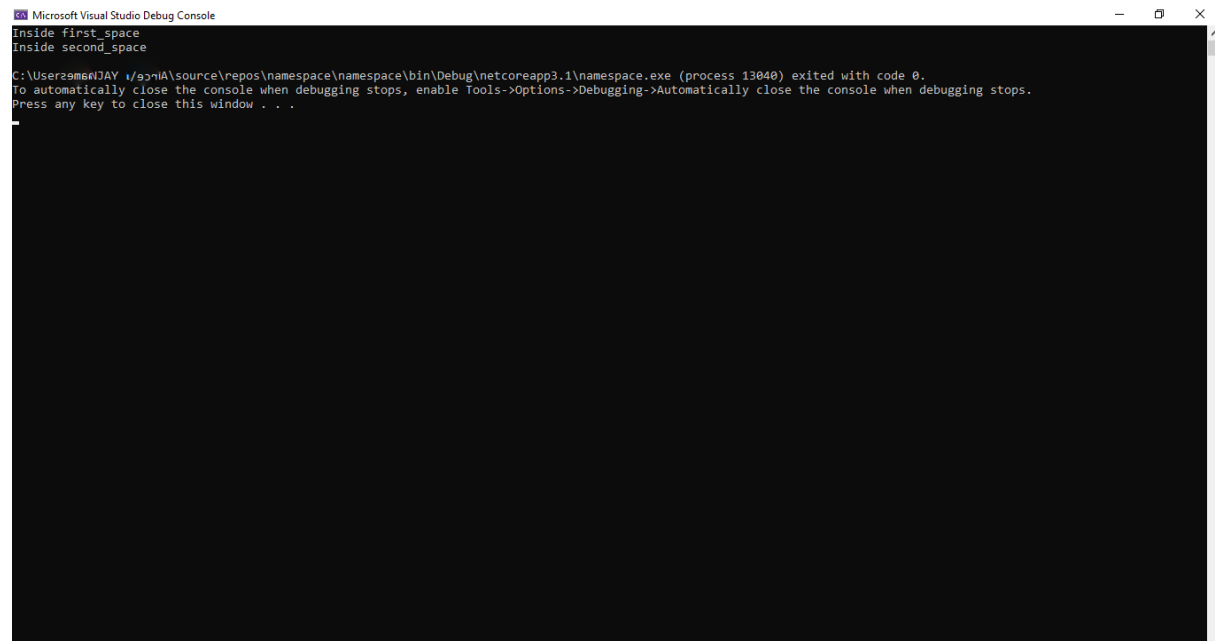
```
1 using System;
2
3
4 namespace first_space
5 {
6     2 references
7     class namespace_cl
8     {
9         1 reference
10        public void func()
11        {
12            Console.WriteLine("Inside first_space");
13        }
14    }
15
16 namespace second_space
17 {
18     2 references
19     class namespace_cl
20     {
21         1 reference
22        public void func()
23        {
24            Console.WriteLine("Inside second_space");
25        }
26    }
27
28 0 references
29 class TestClass
30 {
31     0 references
32     static void Main(string[] args)
33     {
34         first_space.namespace_cl fc = new first_space.namespace_cl();
35         second_space.namespace_cl sc = new second_space.namespace_cl();
36         fc.func();
37         sc.func();
38         Console.ReadKey();
39     }
40 }
```



This screenshot shows the same program after modification. The `Main` method in `TestClass` has been updated to call `func()` on the objects and then `Console.ReadKey()` to keep the console window open. The status bar still shows 'No issues found'.

```
14
15
16 namespace second_space
17 {
18     2 references
19     class namespace_cl
20     {
21         1 reference
22        public void func()
23        {
24            Console.WriteLine("Inside second_space");
25        }
26    }
27
28 0 references
29 class TestClass
30 {
31     0 references
32     static void Main(string[] args)
33     {
34         first_space.namespace_cl fc = new first_space.namespace_cl();
35         second_space.namespace_cl sc = new second_space.namespace_cl();
36         fc.func();
37         sc.func();
38         Console.ReadKey();
39     }
40 }
```

Output:



```
Microsoft Visual Studio Debug Console
inside first_space
inside second_space

C:\Users\amrj\source\repos\namespace\namespace\bin\Debug\netcoreapp3.1\namespace.exe (process 13040) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

