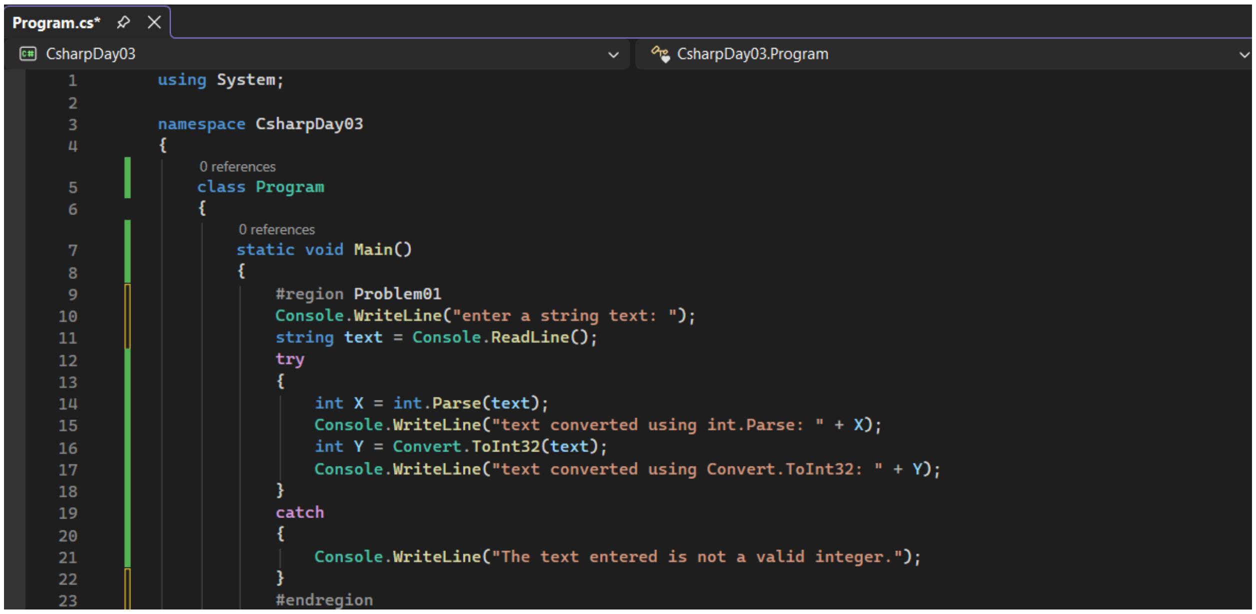
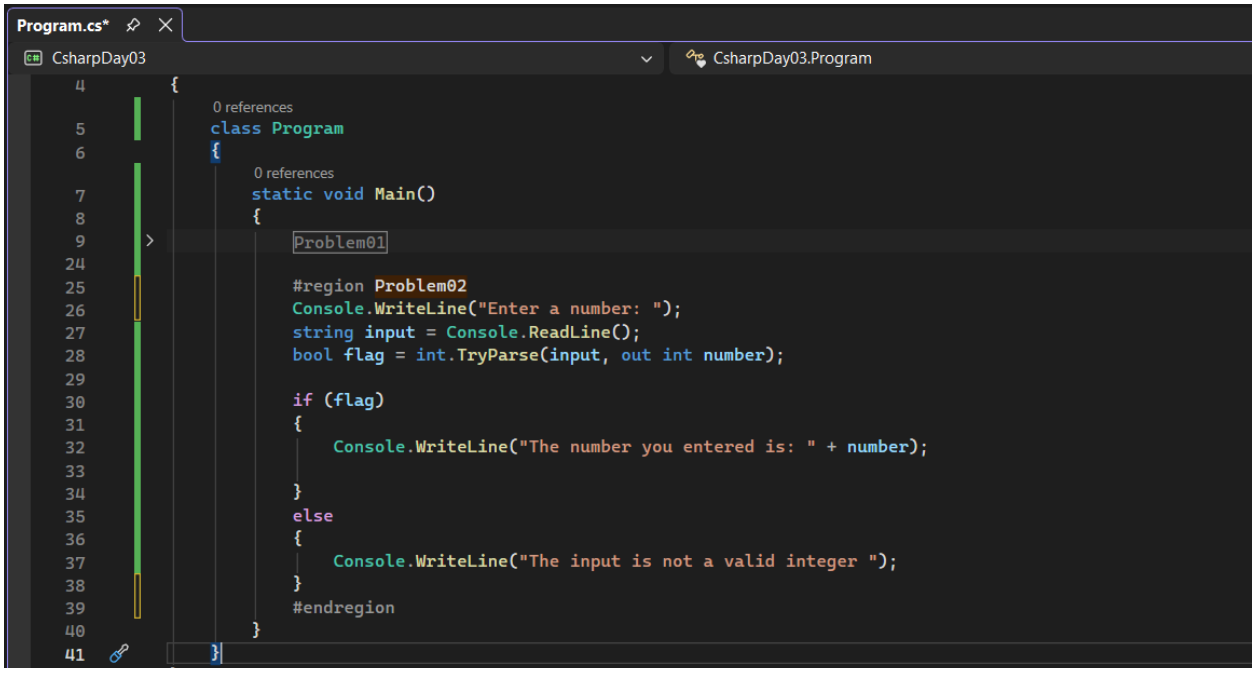


```
using System;

namespace CsharpDay03
{
    class Program
    {
        static void Main()
        {
            #region Problem01
            Console.WriteLine("enter a string text: ");
            string text = Console.ReadLine();
            try
            {
                int X = int.Parse(text);
                Console.WriteLine("text converted using int.Parse: " + X);
                int Y = Convert.ToInt32(text);
                Console.WriteLine("text converted using Convert.ToInt32: " + Y);
            }
            catch
            {
                Console.WriteLine("The text entered is not a valid integer.");
            }
            #endregion
```

Question:

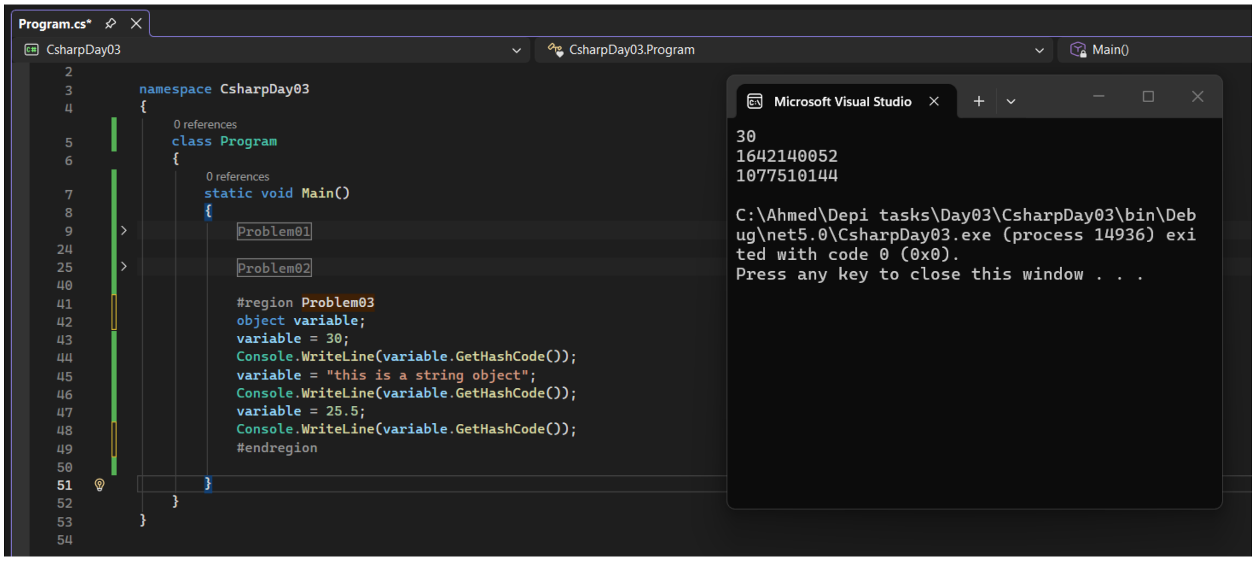
In int.Parse when handling null it will throw an exception while in Convert.ToInt32 it will return 0.



```
{
    class Program
    {
        static void Main()
        {
            Problem01

            #region Problem02
            Console.WriteLine("Enter a number: ");
            string input = Console.ReadLine();
            bool flag = int.TryParse(input, out int number);

            if (flag)
            {
                Console.WriteLine("The number you entered is: " + number);

            }
            else
            {
                Console.WriteLine("The input is not a valid integer ");
            }
            #endregion
        }
    }
```

Question:

Because TryParse handles invalid input when entered by the user without throwing exceptions which improves the overall performance of the code.

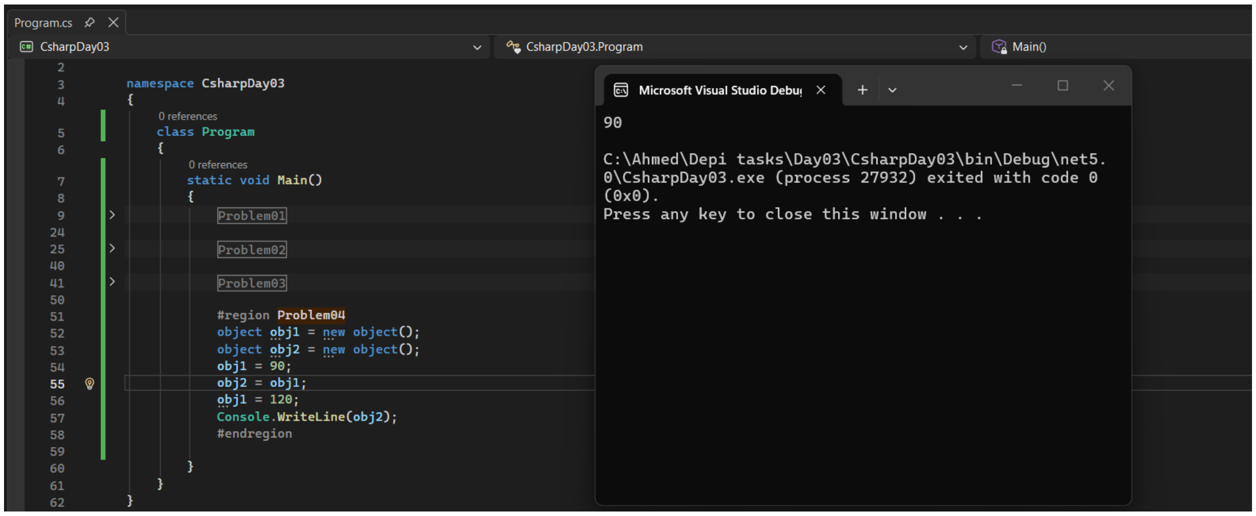


```
namespace CsharpDay03
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            Problem01

            Problem02

            #region Problem03
            object variable;
            variable = 30;
            Console.WriteLine(variable.GetHashCode());
            variable = "this is a string object";
            Console.WriteLine(variable.GetHashCode());
            variable = 25.5;
            Console.WriteLine(variable.GetHashCode());
            #endregion

        }
    }
}
```

```
30
1642140052
1077510144

C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Deb
ug\net5.0\CsharpDay03.exe (process 14936) exi
ted with code 0 (0x0).
Press any key to close this window . . .
```

Question:

It returns an integer numbers that represents an object's hash value. It is mainly used store and retrieve objects efficiently by determining their location.

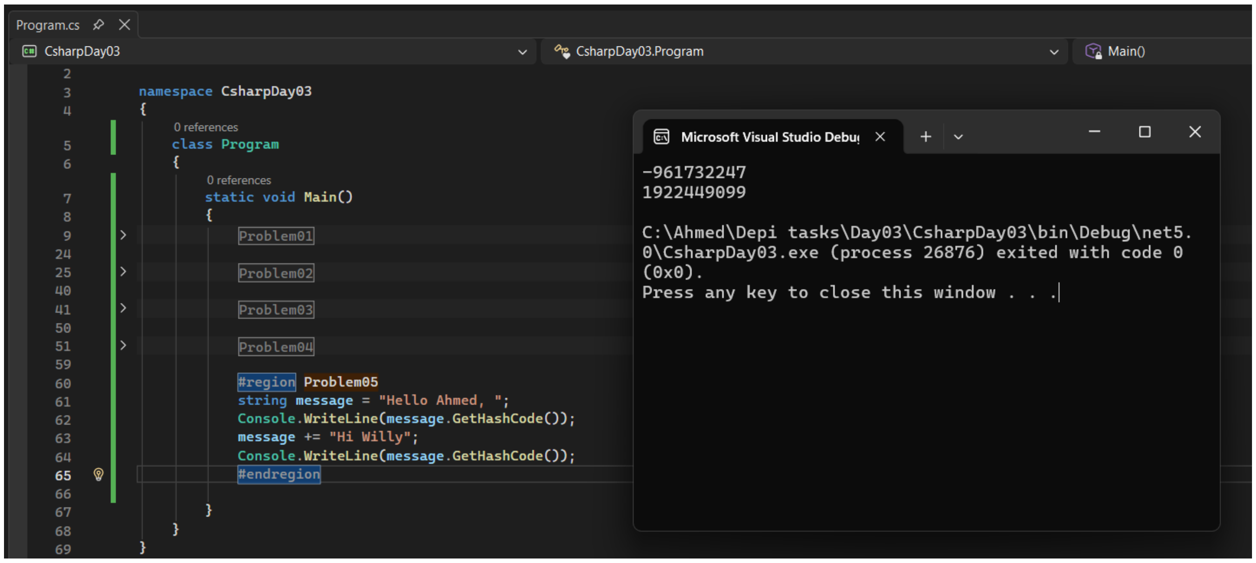


```
namespace CsharpDay03
{
    0 references
    class Program
    {
        0 references
        static void Main()
        {
            Problem01

            Problem02

            Problem03

            #region Problem04
            object obj1 = new object();
            object obj2 = new object();
            obj1 = 90;
            obj2 = obj1;
            obj1 = 120;
            Console.WriteLine(obj2);
            #endregion

        }
    }
}
```

```
90

C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5.
0\CsharpDay03.exe (process 27932) exited with code 0
(0x0).
Press any key to close this window . . .
```

Question:

Reference equality means two variables point to the same object in memory. It is significant because changes made through one reference affect all others, and it helps determine object identity and manage objects efficiently in .NET.

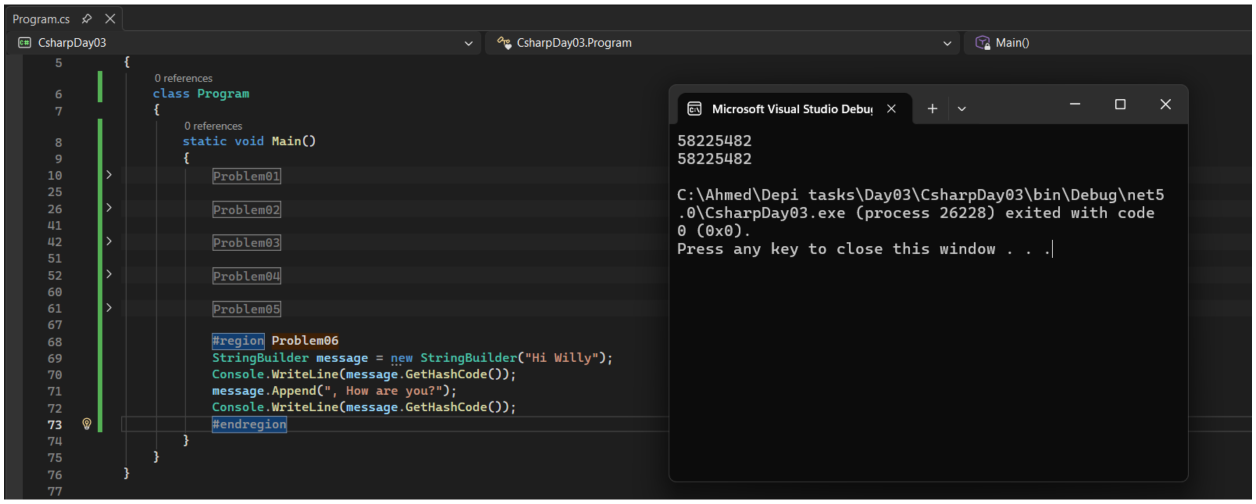


```
namespace CsharpDay03
{
    class Program
    {
        static void Main()
        {
            Problem01

            Problem02

            Problem03

            Problem04

            #region Problem05
            string message = "Hello Ahmed, ";
            Console.WriteLine(message.GetHashCode());
            message += "Hi Willy";
            Console.WriteLine(message.GetHashCode());
            #endregion

        }
    }
}
```

```
-961732247
1922449099

C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5.
0\CsharpDay03.exe (process 26876) exited with code 0
(0x0).
Press any key to close this window . . .
```

Question:

String is immutable because it is made of an array of a fixed size so every time we change the characters of the array it will make a new array of characters even If it is the same size.



```
{
    class Program
    {
        static void Main()
        {
            Problem01

            Problem02

            Problem03

            Problem04

            Problem05

            #region Problem06
            StringBuilder message = new StringBuilder("Hi Willy");
            Console.WriteLine(message.GetHashCode());
            message.Append(", How are you?");
            Console.WriteLine(message.GetHashCode());
            #endregion
        }
    }
}
```

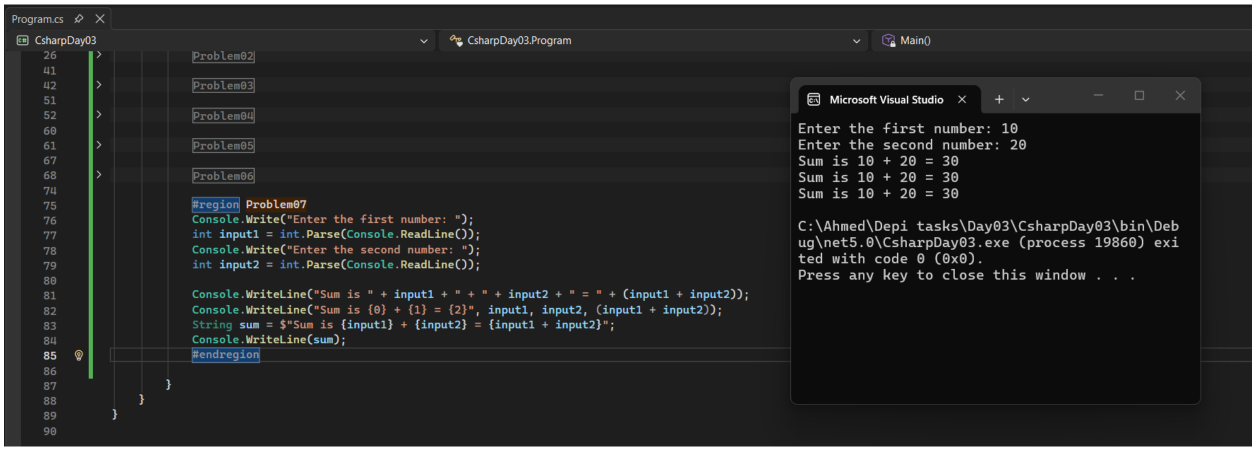
```
58225482
58225482

C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5
.0\CsharpDay03.exe (process 26228) exited with code
0 (0x0).
Press any key to close this window . . .
```

Question:

Because it does not make a new object it modifies the same object that was created before therefor this will make the memory allocation of the objects more efficient.

Question:

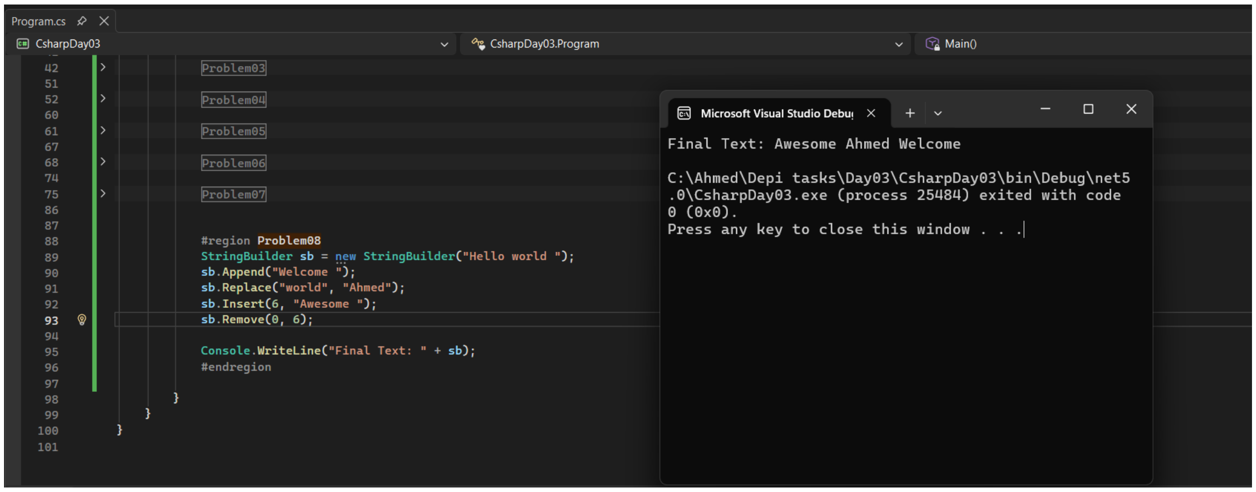
StringBuilder is faster for large-scale string modifications because it modifies the same memory buffer AKA immutable, while string creates a new object every time you change it.



```csharp
            Problem02

            Problem03

            Problem04

            Problem05

            Problem06

            #region Problem07
            Console.Write("Enter the first number: ");
            int input1 = int.Parse(Console.ReadLine());
            Console.Write("Enter the second number: ");
            int input2 = int.Parse(Console.ReadLine());

            Console.WriteLine("Sum is " + input1 + " + " + input2 + " = " + (input1 + input2));
            Console.WriteLine("Sum is {0} + {1} = {2}", input1, input2, (input1 + input2));
            String sum = $"Sum is {input1} + {input2} = {input1 + input2}";
            Console.WriteLine(sum);
            #endregion

        }
    }
}
```

```
Enter the first number: 10
Enter the second number: 20
Sum is 10 + 20 = 30
Sum is 10 + 20 = 30
Sum is 10 + 20 = 30

C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Deb
ug\net5.0\CsharpDay03.exe (process 19860) exi
ted with code 0 (0x0).
Press any key to close this window . . .
```

Question the most used methos is string interpolation because it is clearer also it does not need to make a new object for every word like the concatenation.



```csharp
            Problem03

            Problem04

            Problem05

            Problem06

            Problem07


            #region Problem08
            StringBuilder sb = new StringBuilder("Hello world ");
            sb.Append("Welcome ");
            sb.Replace("world", "Ahmed");
            sb.Insert(6, "Awesome ");
            sb.Remove(0, 6);

            Console.WriteLine("Final Text: " + sb);
            #endregion

        }
    }
}
```

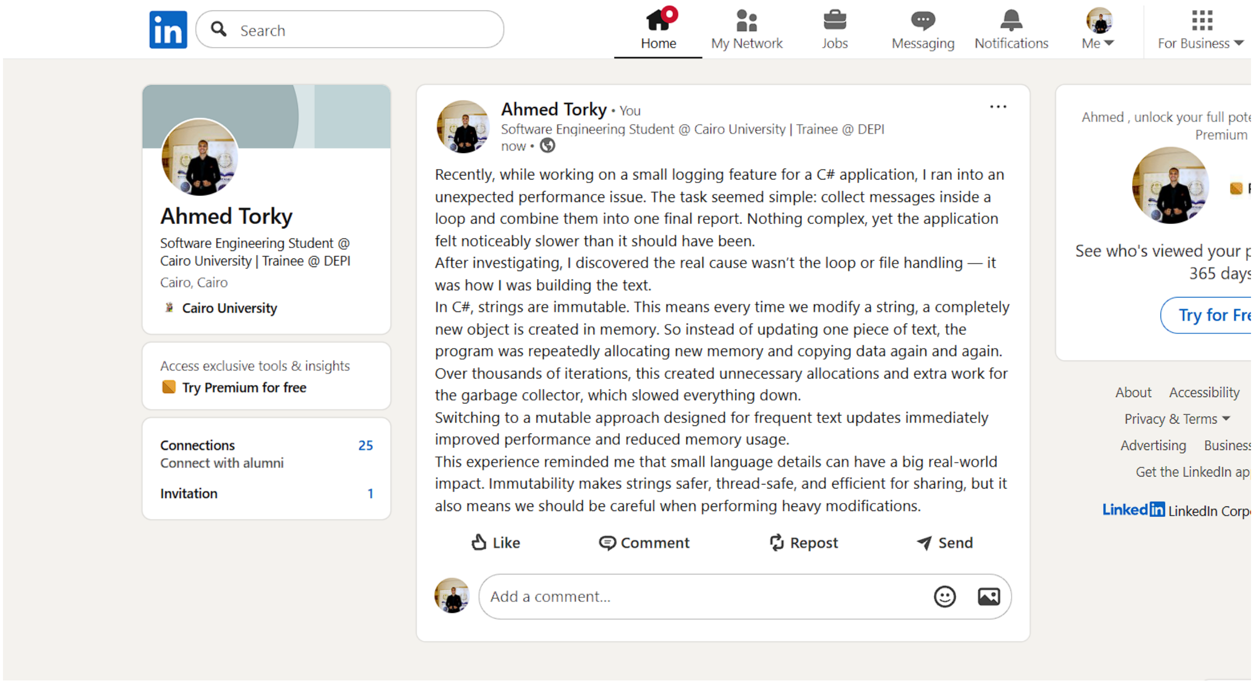
```
Final Text: Awesome Ahmed Welcome

C:\Ahmed\Depi tasks\Day03\CsharpDay03\bin\Debug\net5
.0\CsharpDay03.exe (process 25484) exited with code
0 (0x0).
Press any key to close this window . . .
```

Question:

Because string is Immutable which means that once the object is created any change will create a new object leading to overhead and garbage collection, while StringBuilder is mutable because it modifies internal character buffer without creating new object.

<u>**Part02**</u>

1-LinkedIn Post:



Ahmed Torky
Software Engineering Student @ Cairo University | Trainee @ DEPI
Cairo, Cairo
Cairo University

Access exclusive tools & insights
Try Premium for free

Connections 25
Connect with alumni
Invitation 1

Ahmed Torky • You
Software Engineering Student @ Cairo University | Trainee @ DEPI
now •

Recently, while working on a small logging feature for a C# application, I ran into an unexpected performance issue. The task seemed simple: collect messages inside a loop and combine them into one final report. Nothing complex, yet the application felt noticeably slower than it should have been.
After investigating, I discovered the real cause wasn't the loop or file handling — it was how I was building the text.
In C#, strings are immutable. This means every time we modify a string, a completely new object is created in memory. So instead of updating one piece of text, the program was repeatedly allocating new memory and copying data again and again. Over thousands of iterations, this created unnecessary allocations and extra work for the garbage collector, which slowed everything down.
Switching to a mutable approach designed for frequent text updates immediately improved performance and reduced memory usage.
This experience reminded me that small language details can have a big real-world impact. Immutability makes strings safer, thread-safe, and efficient for sharing, but it also means we should be careful when performing heavy modifications.

2-What's Enum data type, when is it used? And name three common built_in enums used frequently?

An enum is a value type that represents a set of named constants. It makes code more readable, safe and organized.

We can use enum when:

values are fixed and limited, or you want the code to be more readable, or you want to prevent invalid values.

One of the most common built_in enums is:

1) Day of week.

Example: DayOfWeek today = DayOfWeek.Monday;

2) ConsoleColor

Example: Console.ForegroundColor = ConsoleColor.Green;

2) FileMode

Examples:

FileMode.Open

FileMode.Create

FileMode.Append

3- what are scenarios to use string Vs StringBuilder?

For example when making a few operations like storing names, messages or labels string will be most suitable for these operations.

While if you are making heavy modifications like loops, generating reports or even building JSON/XML StringBuilder will be more faster and memory efficient.

Part03

5-what meant by user defined constructor and its role in initialization.

A user-defined constructor is a constructor that the developer explicitly writes in a class to control how an object is created and initialized.

Its main role is to set the initial state of the object, assign values to variables, or perform any setup required before the object is used.

6- compare between Array and Linked List.

First of all, an array stores elements in contiguous memory, allowing fast access by index, but its size is fixed and inserting or deleting elements in the middle is slow because elements must be shifted.

On the other hand, A linked list stores elements in separate nodes connected by references, allowing dynamic size and fast insertions or deletions, but accessing a specific element is slower because you must traverse the list from the beginning.