

PRE-LAB

1. What are the Advantages of LINQ over SQL?

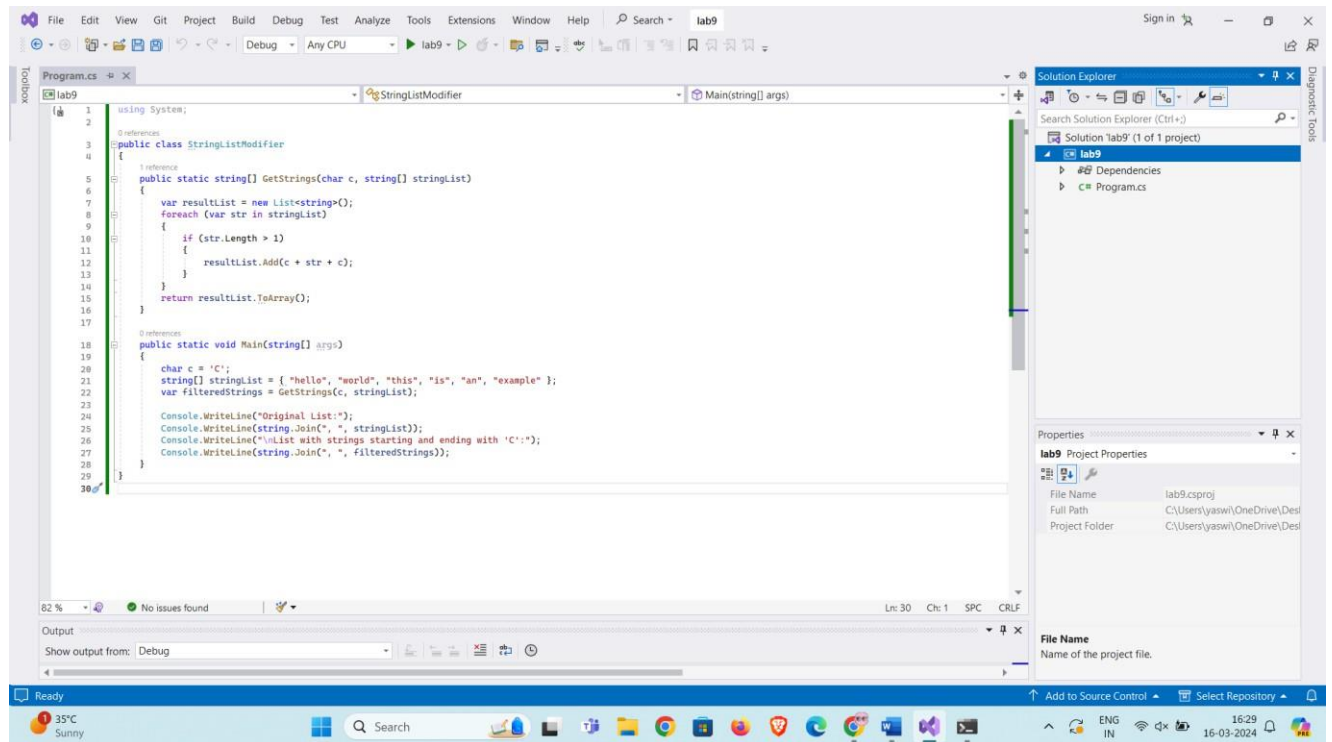
Solution:

1. Integration with Language: LINQ (Language Integrated Query) is integrated into C# and other .NET languages, allowing developers to write queries directly in their familiar programming language syntax. This reduces the need to switch between different languages (e.g., C# and SQL) and improves code readability and maintainability.
2. Compile-time Checking: LINQ queries are checked for syntax and type errors at compile time, providing early error detection and improving code reliability. In contrast, SQL queries are often only checked at runtime, leading to potential errors being discovered later.
3. Query Expressiveness: LINQ offers a rich set of query operators that allow developers to express complex queries in a concise and readable manner. This includes operators for filtering, sorting, grouping, joining, and aggregating data, making it easier to write and understand queries compared to SQL.
4. Strongly Typed Queries: LINQ queries are strongly typed, which means that the compiler can enforce type safety. This helps prevent runtime errors related to data type mismatches and provides better IntelliSense support in IDEs, leading to improved developer productivity.
5. Platform Independence: LINQ is not tied to a specific database platform or technology. It can be used with various data sources such as SQL databases, XML documents, in-memory collections, and more. This makes LINQ more versatile and suitable for a wide range of applications compared to SQL, which is specific to relational databases.

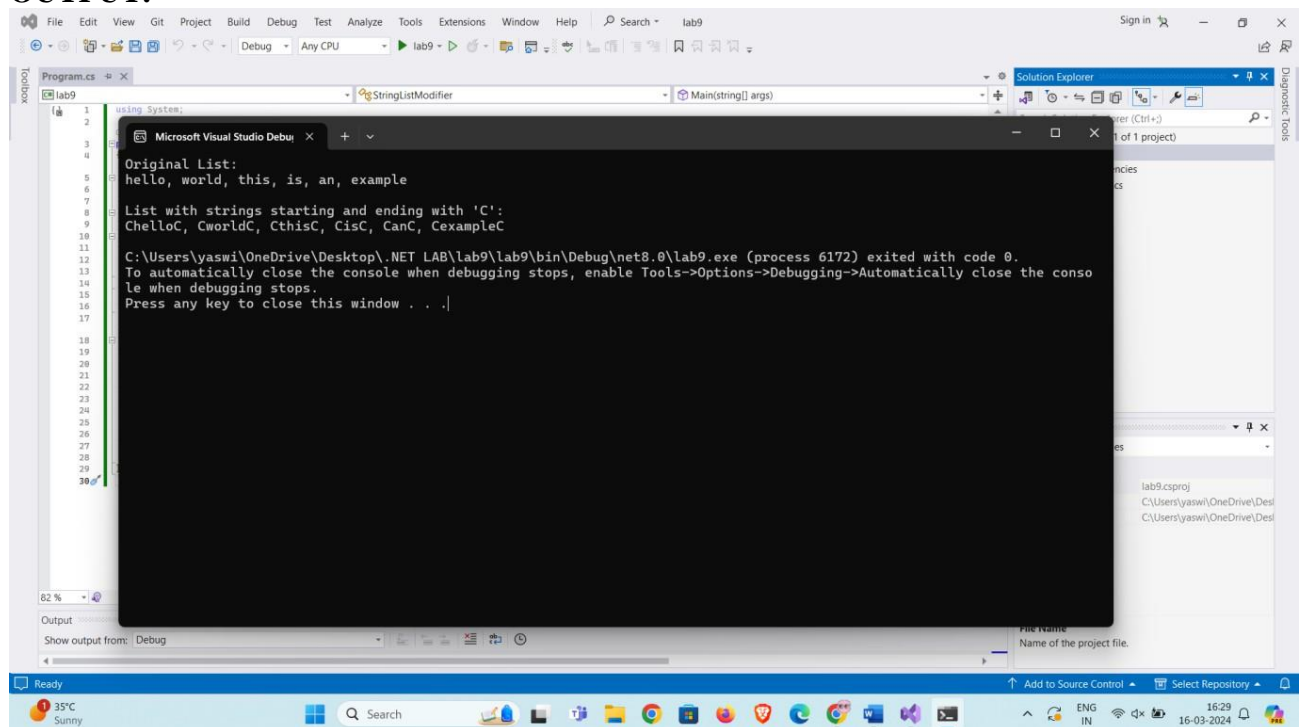
IN-LAB:

At the Low level, you need to solve the following five tasks:

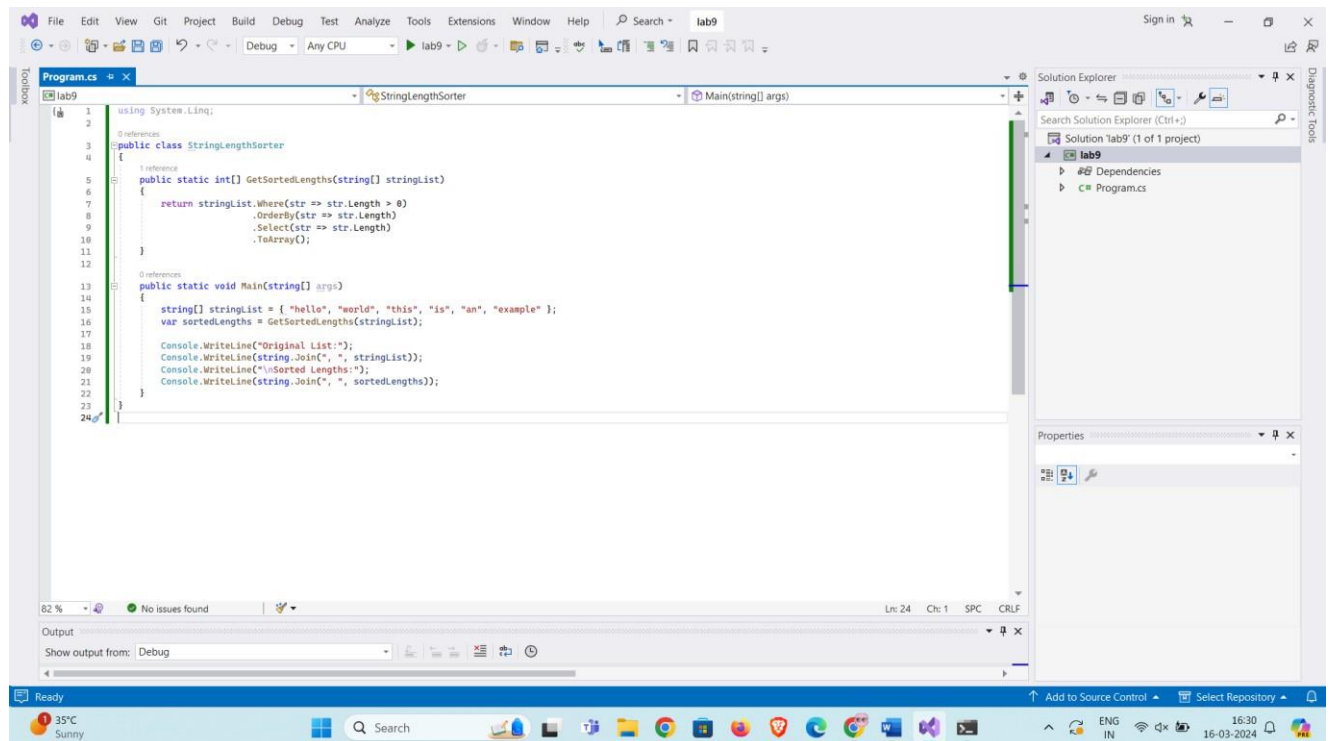
TASK 1: The character **C** and a sequence of non-empty strings **stringList** are given. Get a new sequence of strings with more than one character from the **stringList**, starting and ending with **C**.



OUTPUT:



TASK 2: A sequence of non-empty strings `stringList` is given. Get a sequence of ascending sorted integer values equal to the lengths of the strings included in the `stringList` sequence.

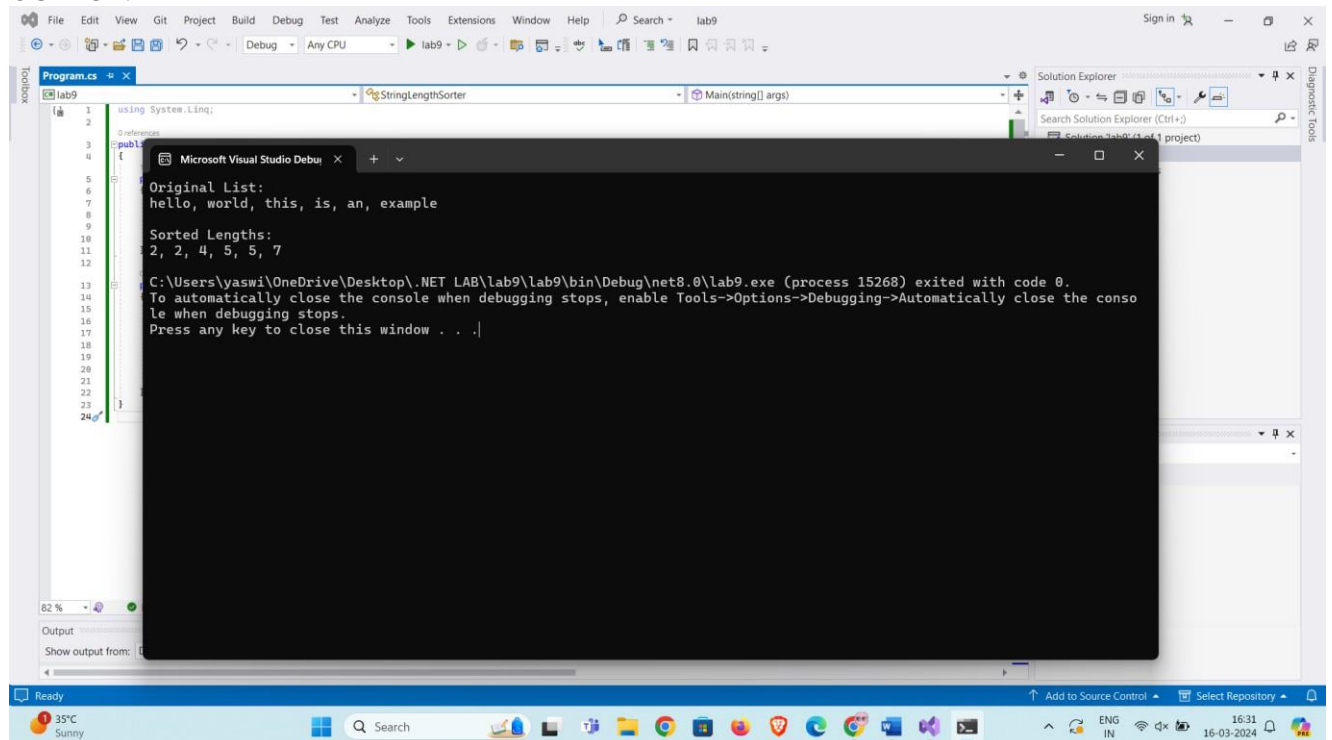


The screenshot shows the Visual Studio IDE with a C# project named 'lab9'. The file 'Program.cs' is open, showing the following code:

```
1 using System.Linq;
2
3 public class StringLengthSorter
4 {
5     public static int[] GetSortedLengths(string[] stringList)
6     {
7         return stringList.Where(str => str.Length > 0)
8             .OrderBy(str => str.Length)
9             .Select(str => str.Length)
10            .ToArray();
11     }
12
13     public static void Main(string[] args)
14     {
15         string[] stringList = { "hello", "world", "this", "is", "an", "example" };
16         var sortedLengths = GetSortedLengths(stringList);
17
18         Console.WriteLine("Original List:");
19         Console.WriteLine(string.Join(", ", stringList));
20         Console.WriteLine("\nSorted Lengths:");
21         Console.WriteLine(string.Join(", ", sortedLengths));
22     }
23 }
24
```

The Solution Explorer on the right shows the project 'lab9' with a file 'Program.cs'. The Properties window is empty. The status bar at the bottom indicates '82 %' and 'No issues found'.

OUTPUT:



The screenshot shows the Visual Studio IDE with the same code as the previous image. A 'Microsoft Visual Studio Debug Console' window is open, displaying the output of the program:

```
Original List:
hello, world, this, is, an, example

Sorted Lengths:
2, 2, 4, 5, 5, 7

C:\Users\yaswi\OneDrive\Desktop\NET LAB\lab9\lab9\bin\Debug\net8.0\lab9.exe (process 15268) 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 status bar at the bottom indicates '82 %' and 'No issues found'.

TASK 3: A sequence of non-empty strings `stringList` is given. Get a new sequence of strings, where each string consists of the first and last characters of the corresponding string in the `stringList` sequence.

```

1  public class FirstLastCharExtractor
2  {
3      1 reference
4      public static string[] GetFirstLastChars(string[] stringList)
5      {
6          var resultList = new List<string>();
7          foreach (var str in stringList)
8          {
9              if (str.Length > 1)
10             {
11                 resultList.Add(str.Substring(0, 1) + str.Substring(str.Length - 1));
12             }
13         }
14         return resultList.ToArray();
15     }
16
17     0 references
18     public static void Main(string[] args)
19     {
20         string[] stringList = { "hello", "world", "this", "is", "an", "example" };
21         var firstLastChars = GetFirstLastChars(stringList);
22
23         Console.WriteLine("Original List:");
24         Console.WriteLine(string.Join(", ", stringList));
25         Console.WriteLine("\nList with First and Last Characters:");
26         Console.WriteLine(string.Join(", ", firstLastChars));
27     }

```

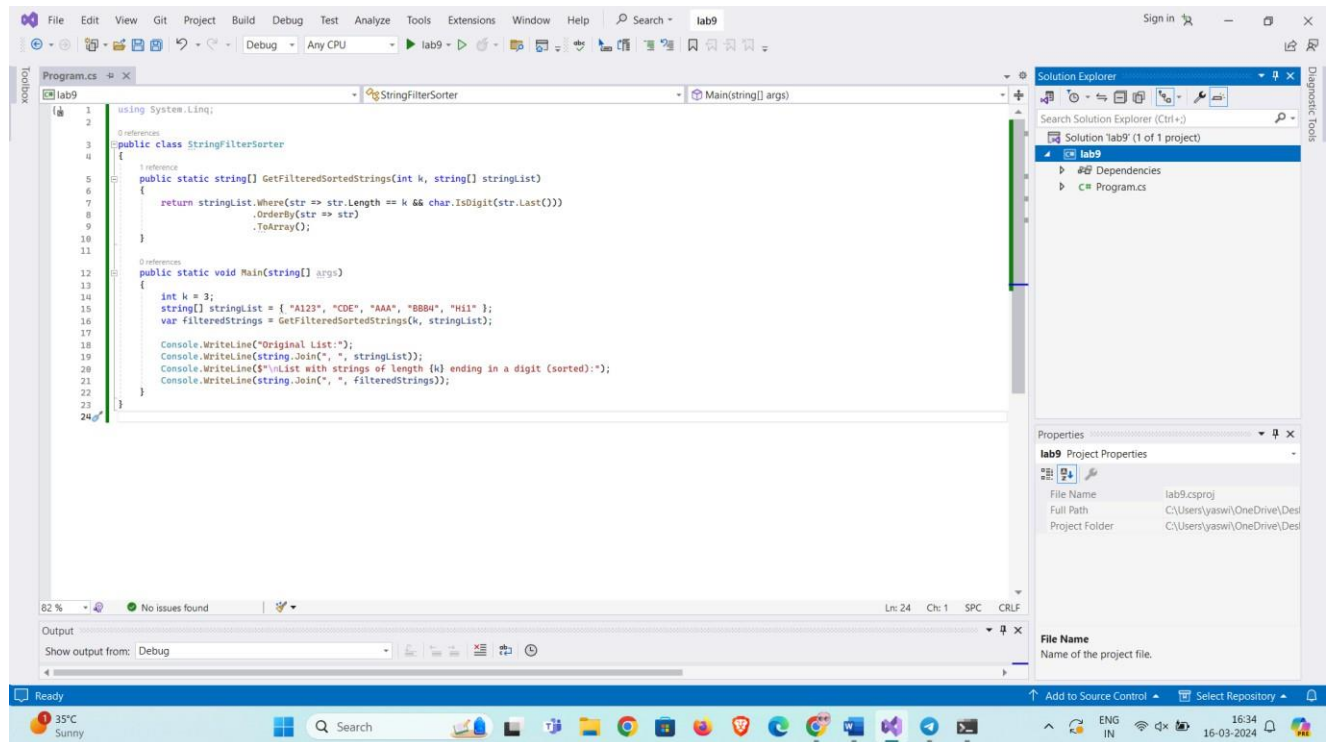
OUTPUT:

```

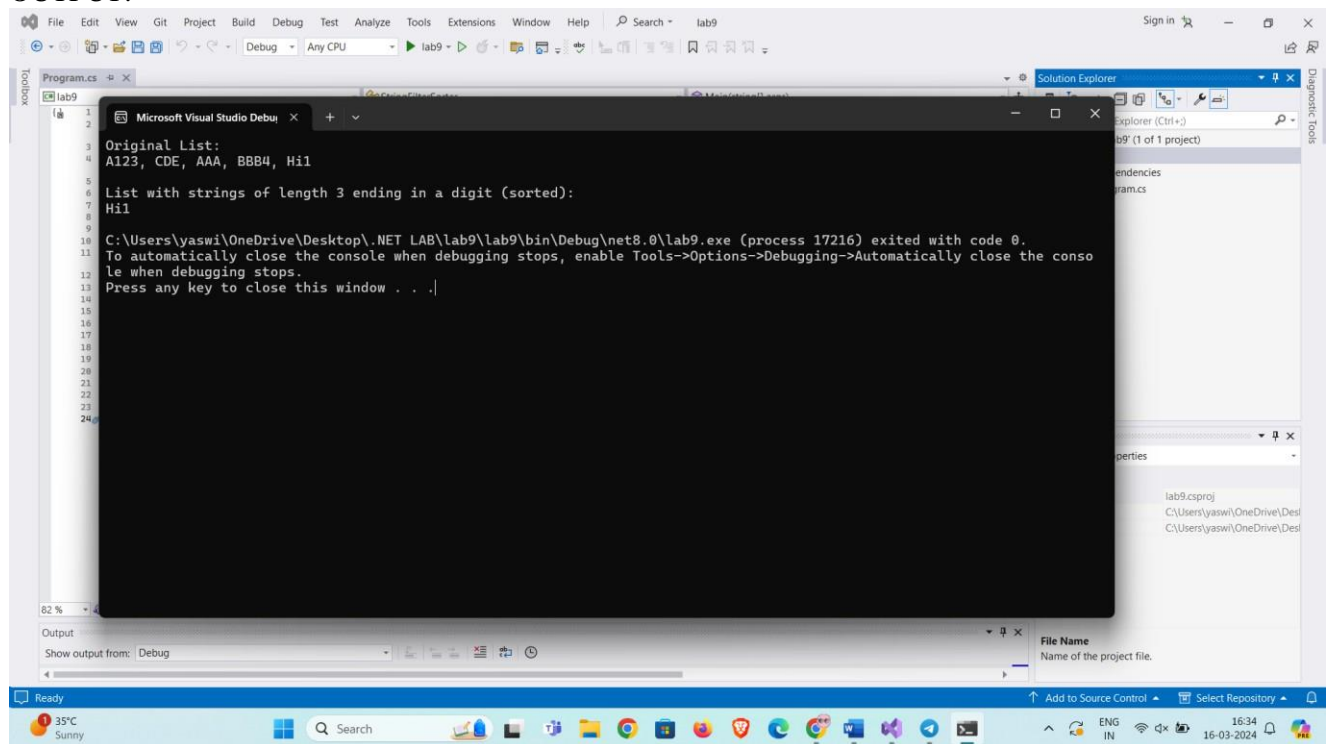
1  Original List:
2  hello, world, this, is, an, example
3
4  List with First and Last Characters:
5  ho, wd, ts, is, an, ee
6
7  C:\Users\yasw1\OneDrive\Desktop\ .NET LAB\lab9\lab9\bin\Debug\net8.0\lab9.exe (process 15956) exited with code 0.
8  To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
9  Press any key to close this window . . .

```

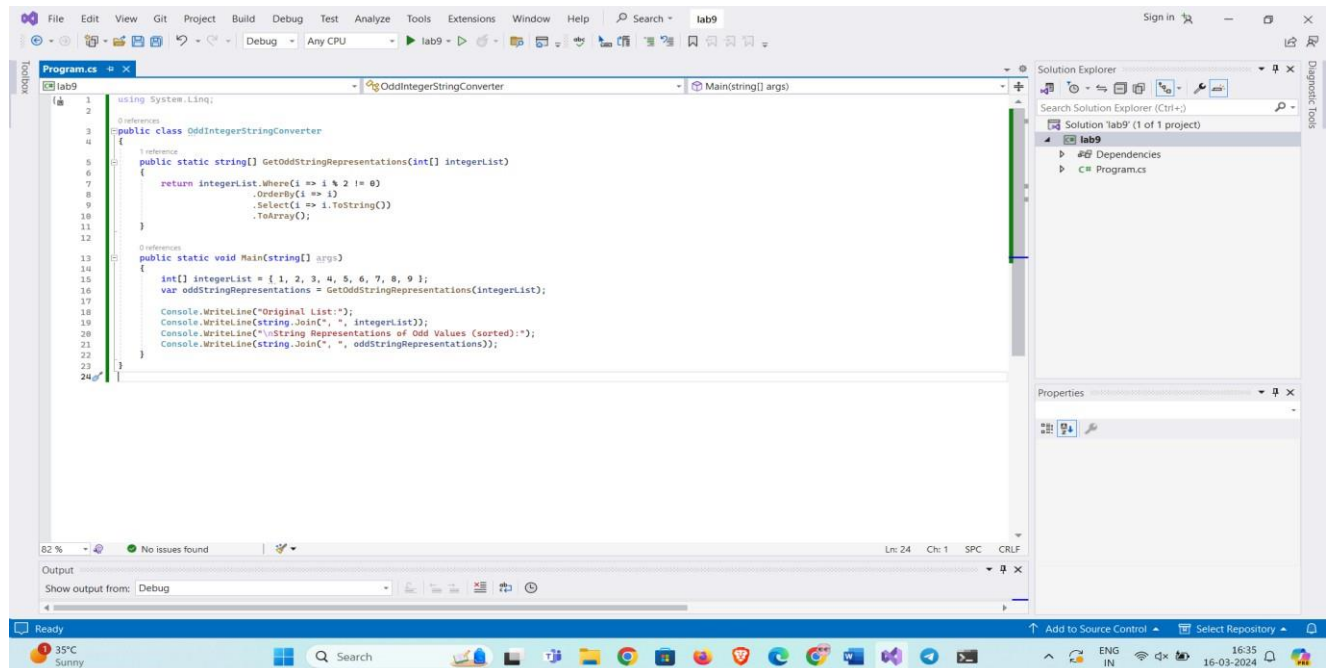
TASK 4: A positive integer **K** and a sequence of non-empty strings `stringList` are given. Strings of the sequence contain only numbers and capital letters of the Latin alphabet. Get from `stringList` all strings of length **K** ending in a digit and sort them in ascending order.



OUTPUT:

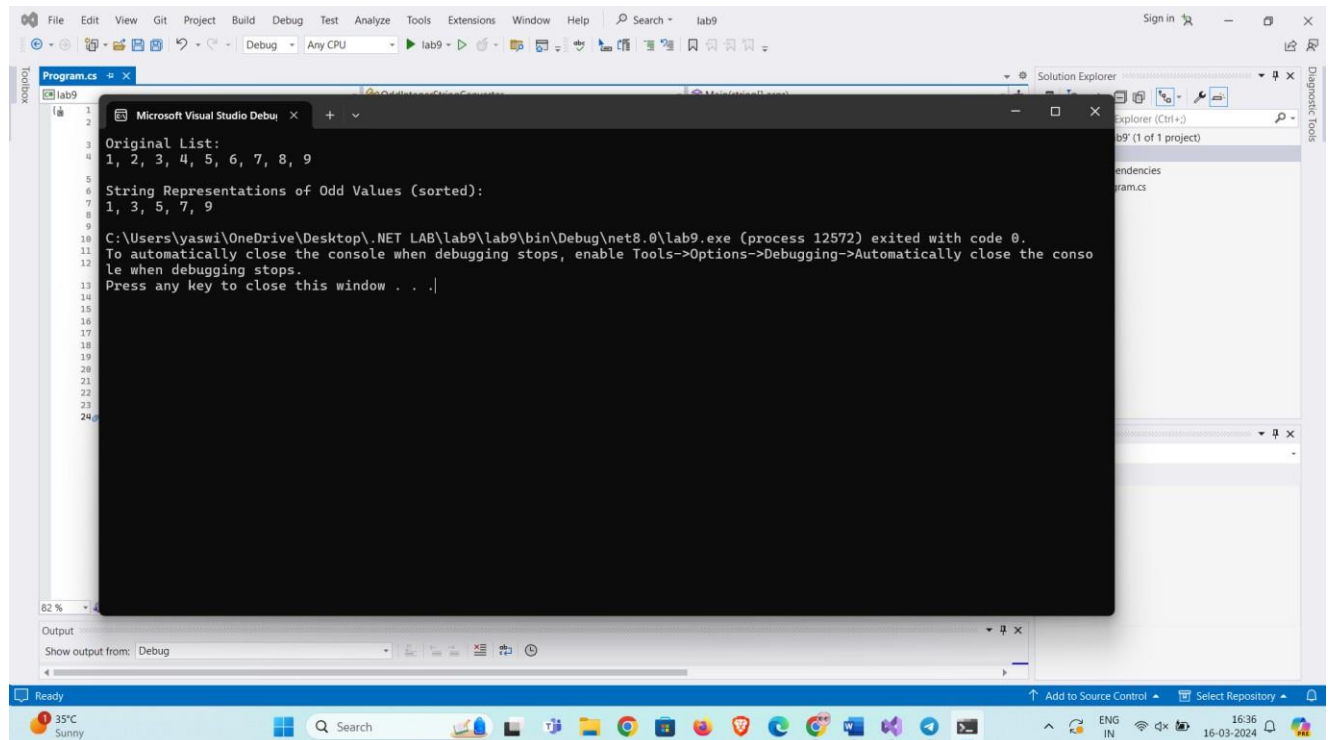


TASK 5: A sequence of positive integer values integerList is given. Get sequence of string representations of only odd integerList values and sort in ascending order.



```
1 using System.Linq;
2
3 public class OddIntegerStringConverter
4 {
5     public static string[] GetOddStringRepresentations(int[] integerList)
6     {
7         return integerList.Where(i => i % 2 != 0)
8             .OrderBy(i => i)
9             .Select(i => i.ToString())
10            .ToArray();
11     }
12
13     public static void Main(string[] args)
14     {
15         int[] integerList = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
16         var oddStringRepresentations = GetOddStringRepresentations(integerList);
17
18         Console.WriteLine("Original List:");
19         Console.WriteLine(string.Join(", ", integerList));
20         Console.WriteLine("String Representations of Odd Values (sorted):");
21         Console.WriteLine(string.Join(", ", oddStringRepresentations));
22     }
23 }
24
```

OUTPUT:



```
Original List:
1, 2, 3, 4, 5, 6, 7, 8, 9
String Representations of Odd Values (sorted):
1, 3, 5, 7, 9
C:\Users\yaswi\OneDrive\Desktop\ .NET LAB\lab9\lab9\bin\Debug\net8.0\lab9.exe (process 12572) 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 . . .
```

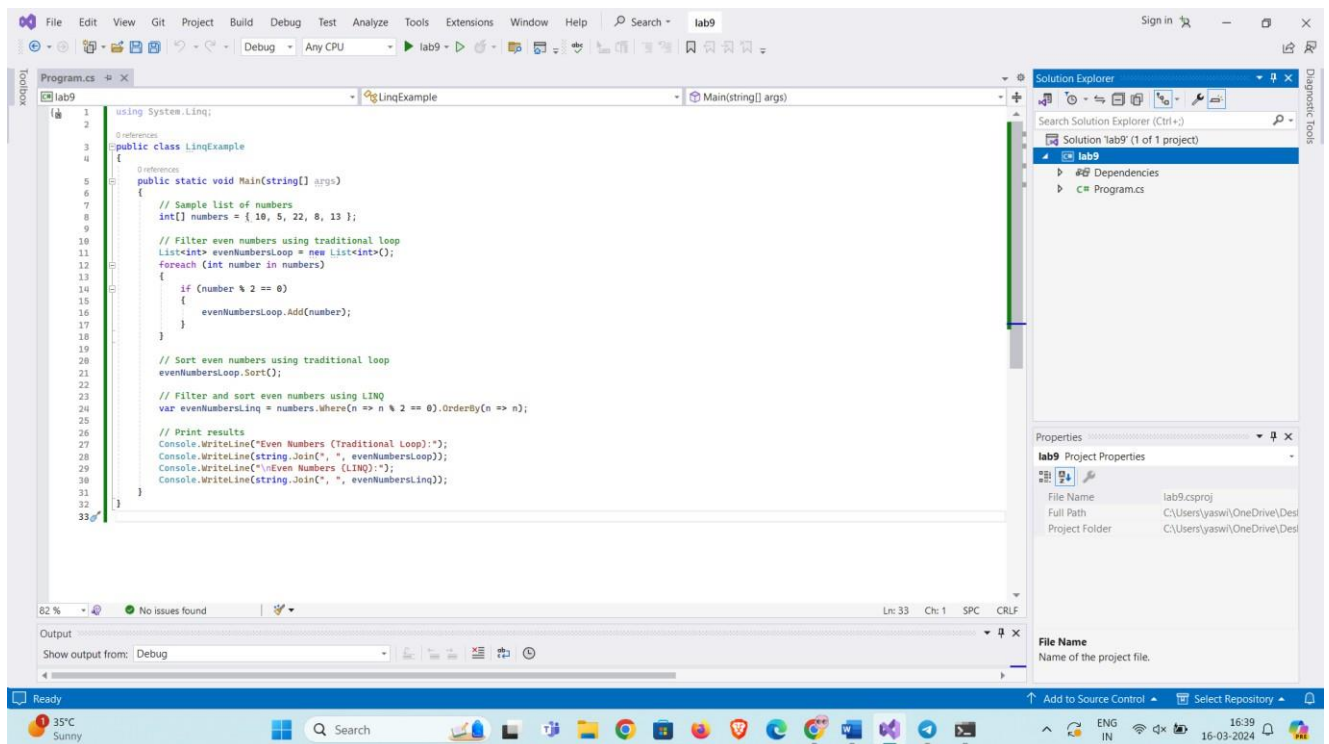
POST-LAB

1. What is necessity of using LINQ? Explain with Suitable example?

Solution: Necessity of Using LINQ:

LINQ (Language Integrated Query) provides a powerful and concise way to query and manipulate data in C#. Here's why it's important:

1. **Readability:** LINQ statements resemble SQL queries, making code easier to read and understand, especially for developers familiar with SQL.
2. **Maintainability:** LINQ code is often more concise than traditional loops and conditional statements, reducing complexity and improving maintainability.
3. **Type Safety:** LINQ leverages strong typing in C#, preventing errors related to data types during compilation.
4. **Expressive:** LINQ allows chaining multiple operations (filtering, sorting, etc.) into a single statement, improving code expressiveness.
5. **Integration with C#:** LINQ is seamlessly integrated with the C# language, allowing for a cohesive programming experience.



OUTPUT:

