

PRE-LAB

1.What are the Advantages of LINQ over SQL?

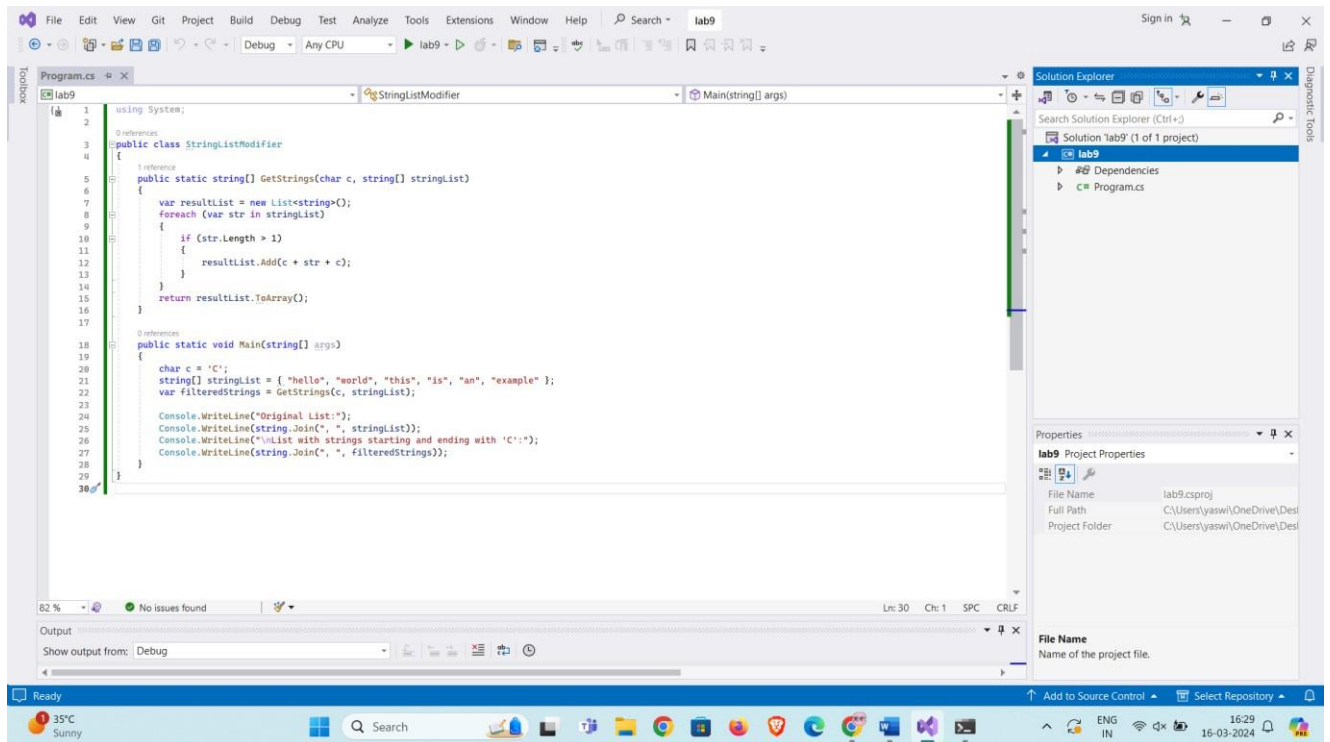
Solution:

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.

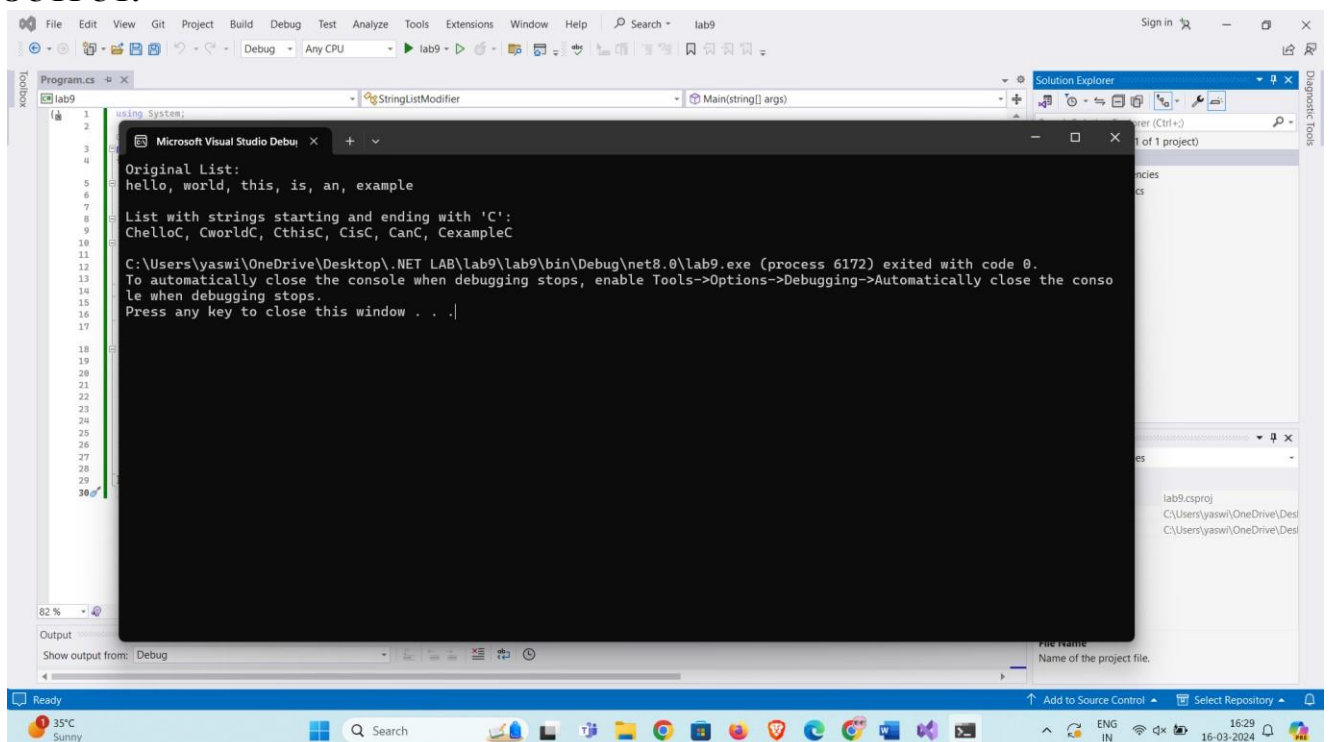
IN-LAB:

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

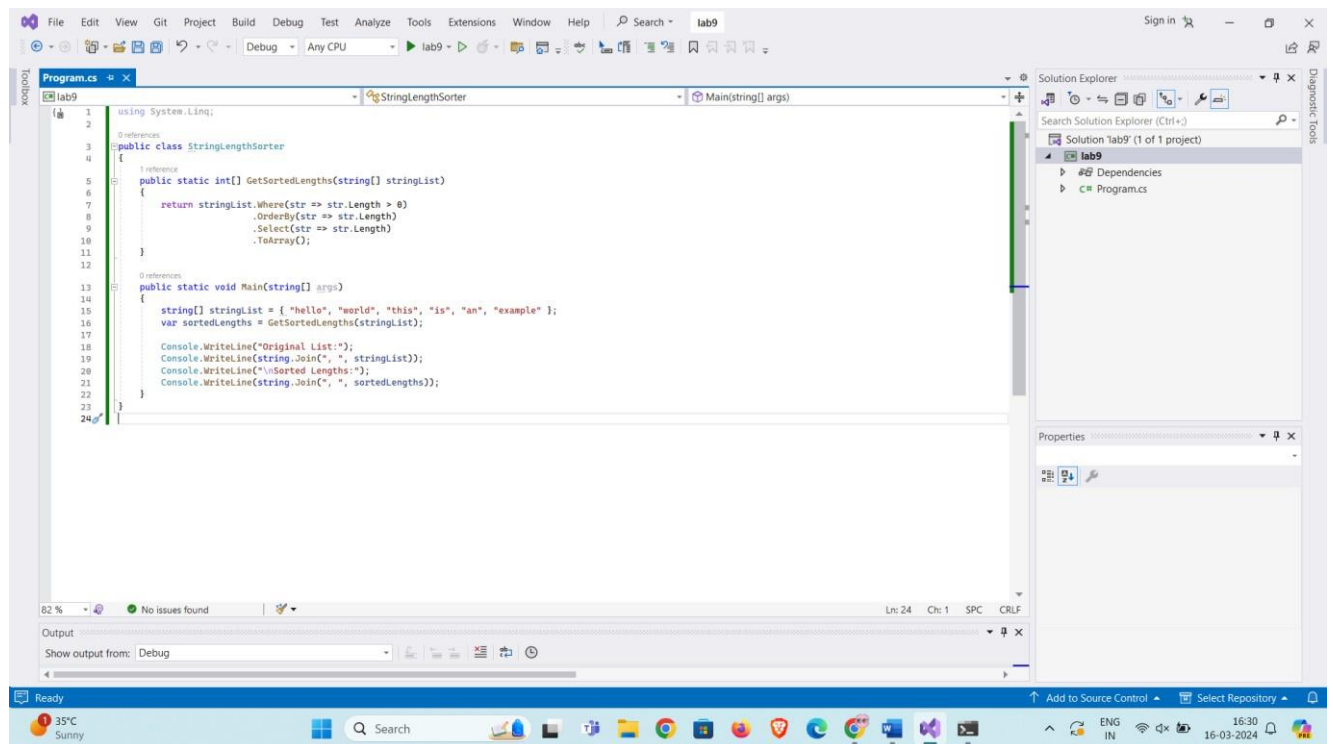
TASK 1: The character **C** and a sequence of non-empty strings **stringList** are given. Get a new sequence of strings with more than one character from the **stringList**, starting and ending with **C**.



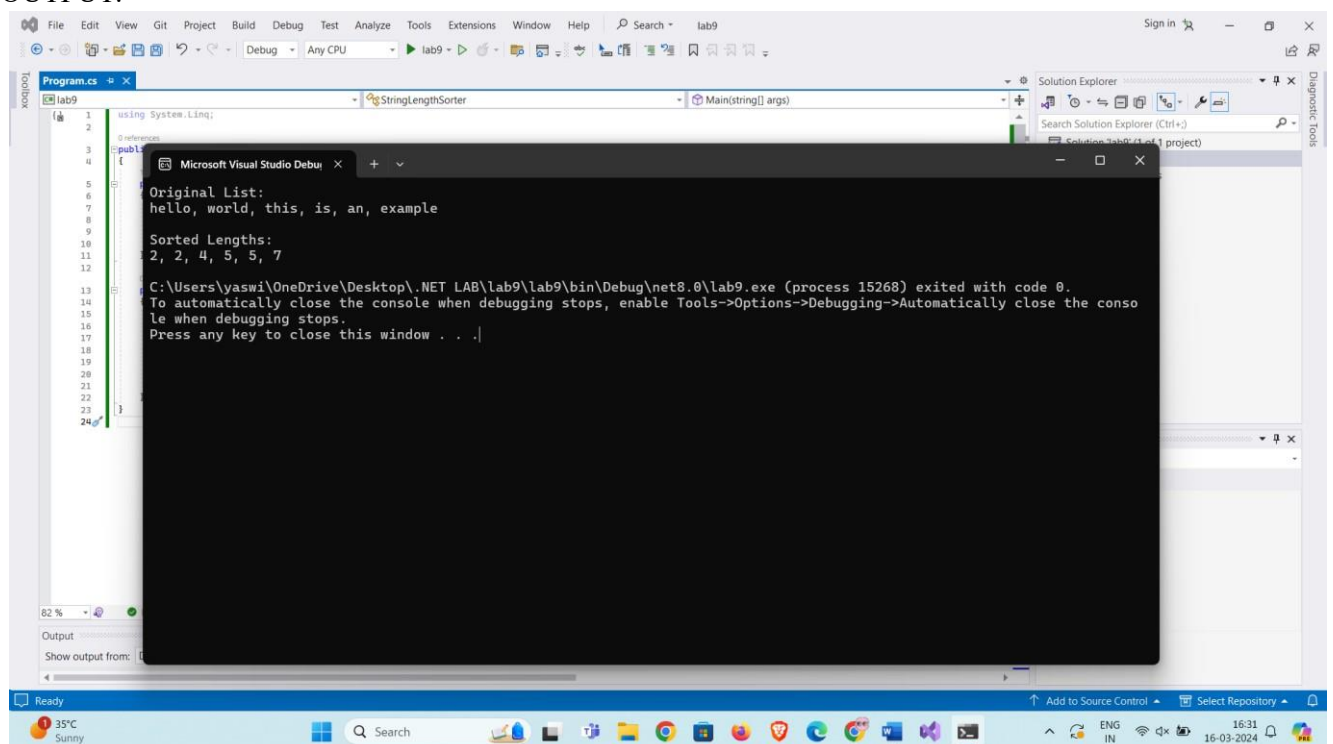
OUTPUT:



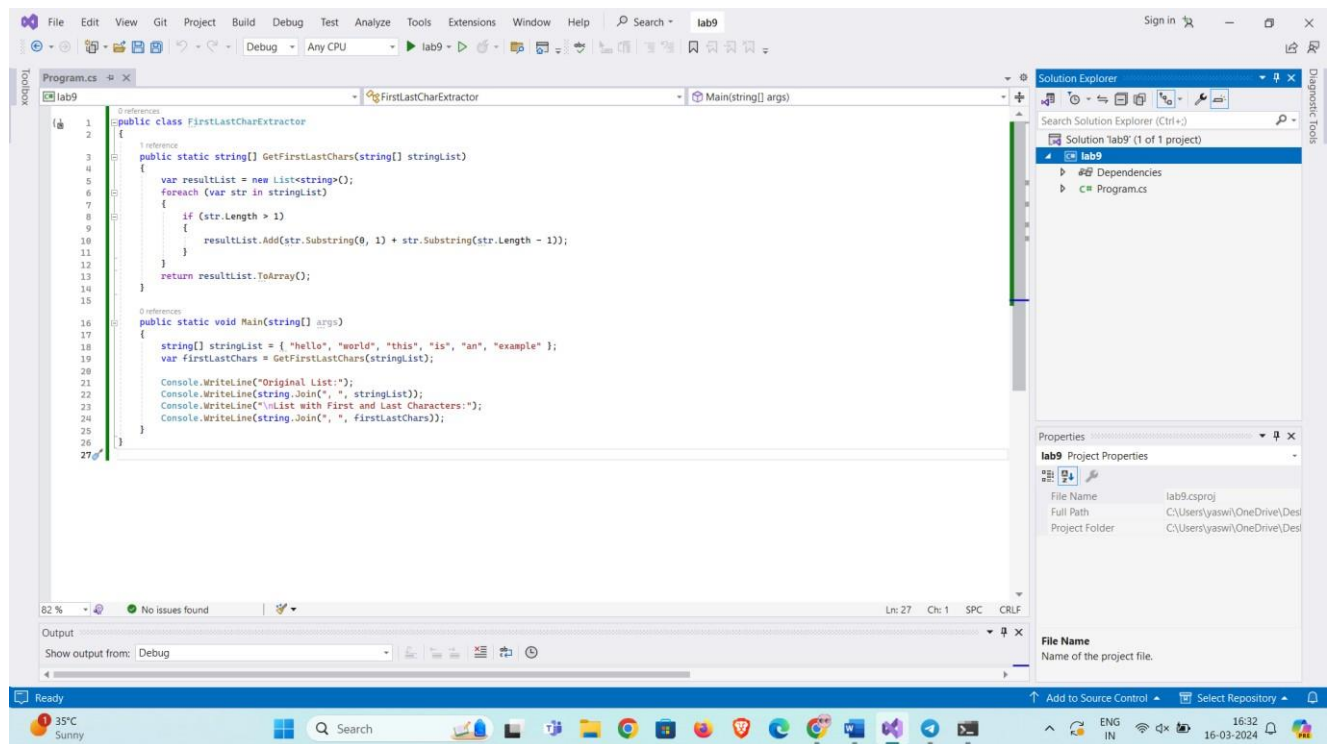
TASK 2: A sequence of non-empty strings `stringList` is given. Get a sequence of ascending sorted integer values equal to the lengths of the strings included in the `stringList` sequence.



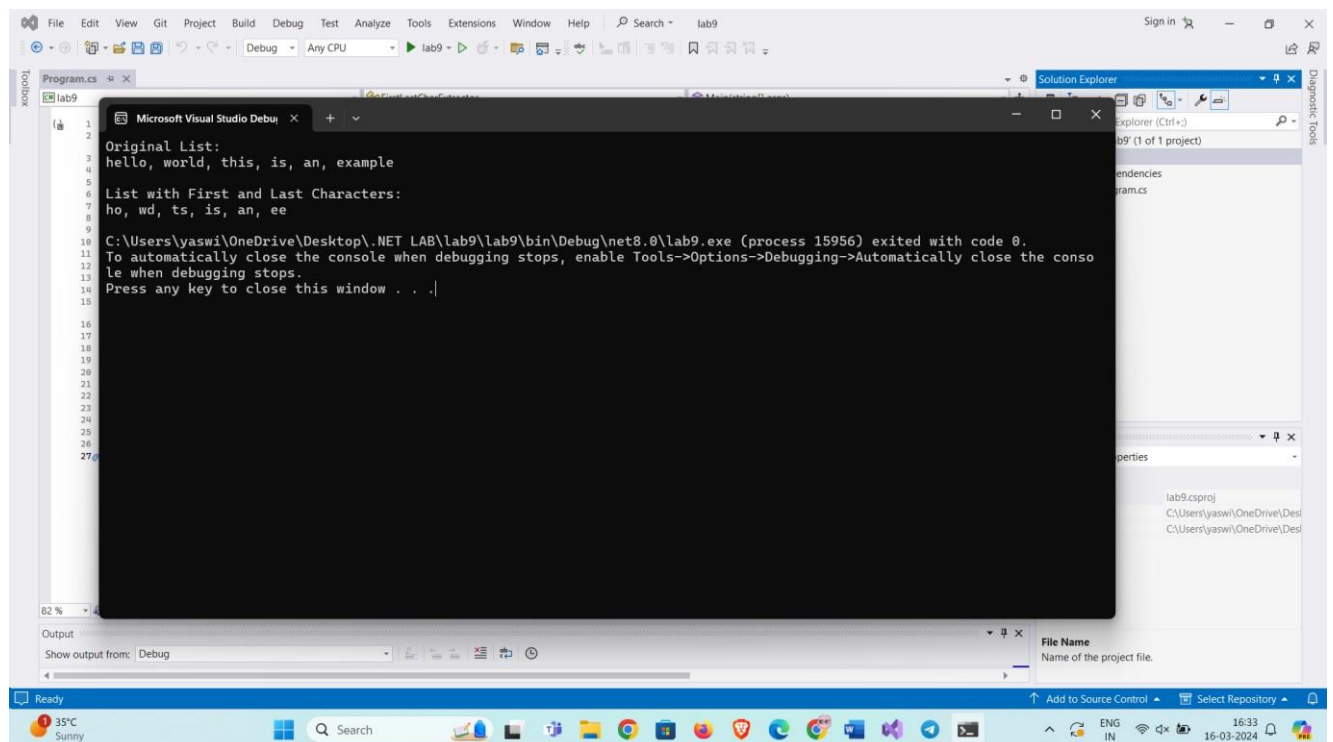
OUTPUT:



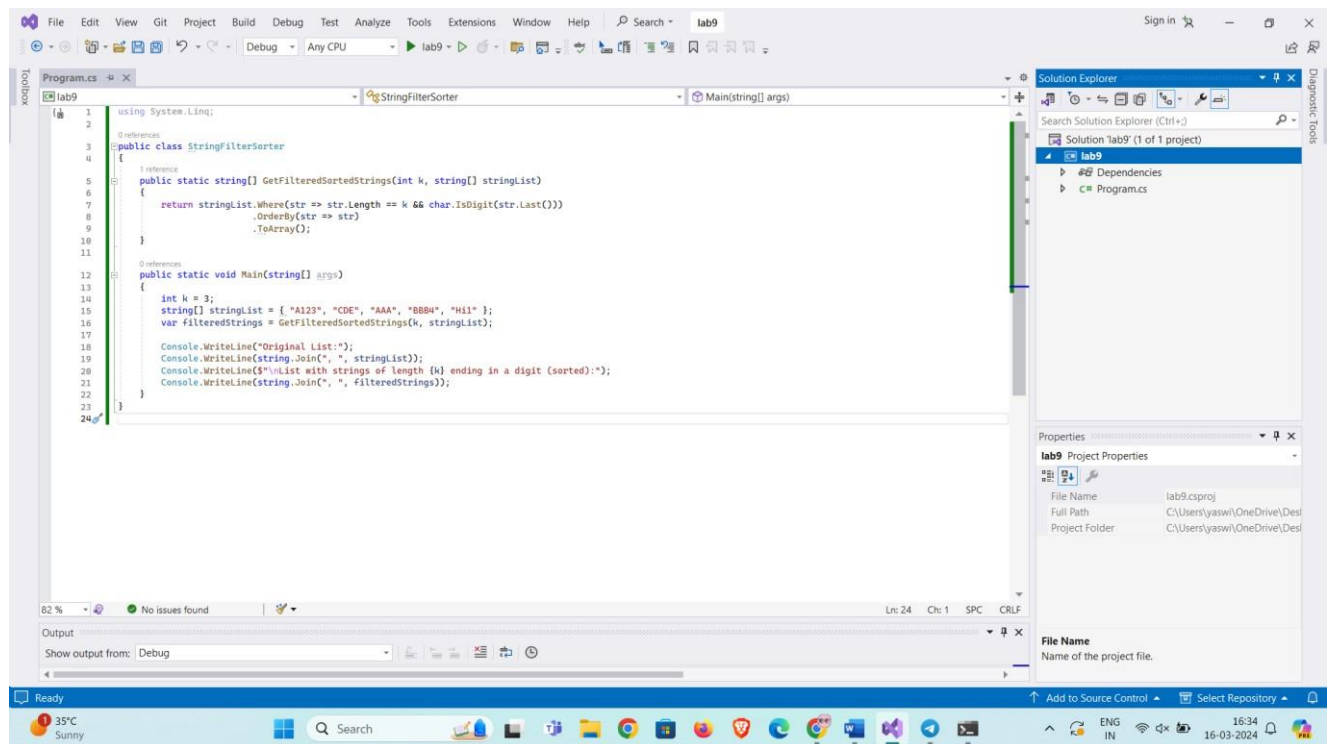
TASK 3: A sequence of non-empty strings `stringList` is given. Get a new sequence of strings, where each string consists of the first and last characters of the corresponding string in the `stringList` sequence.



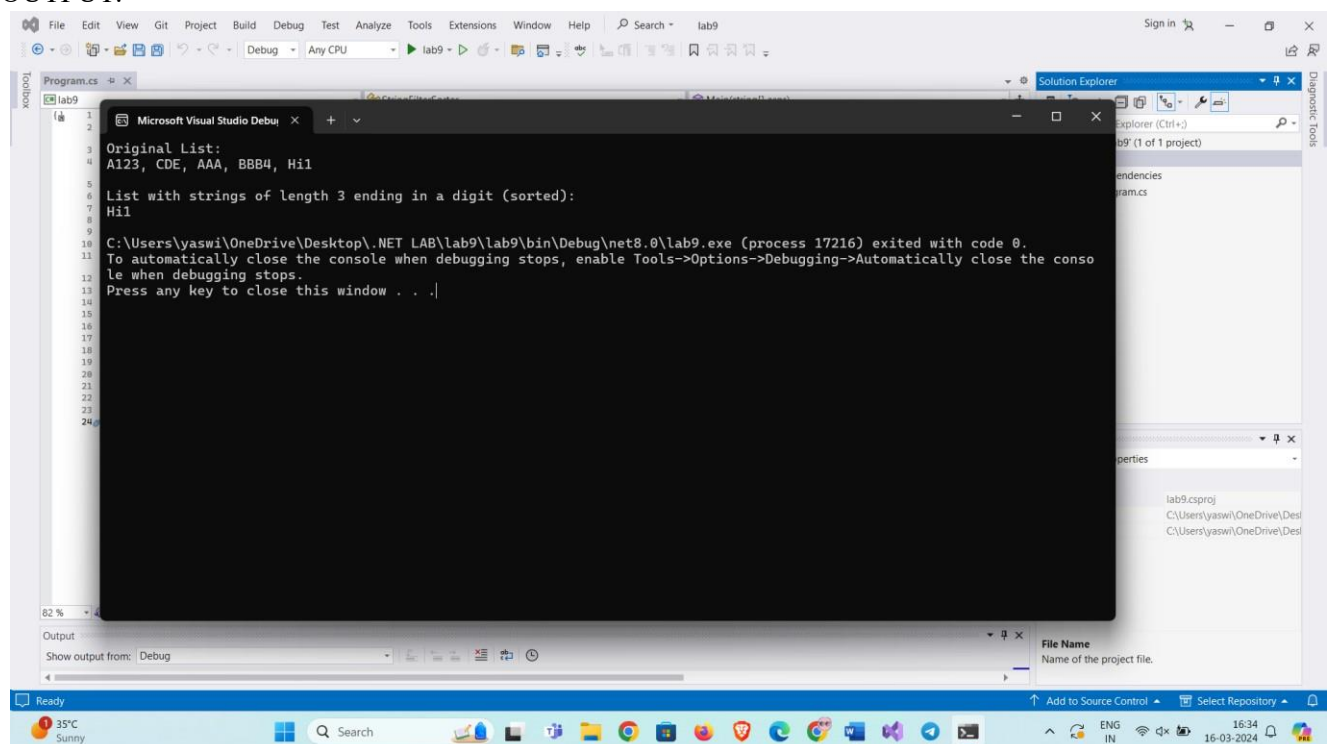
OUTPUT:



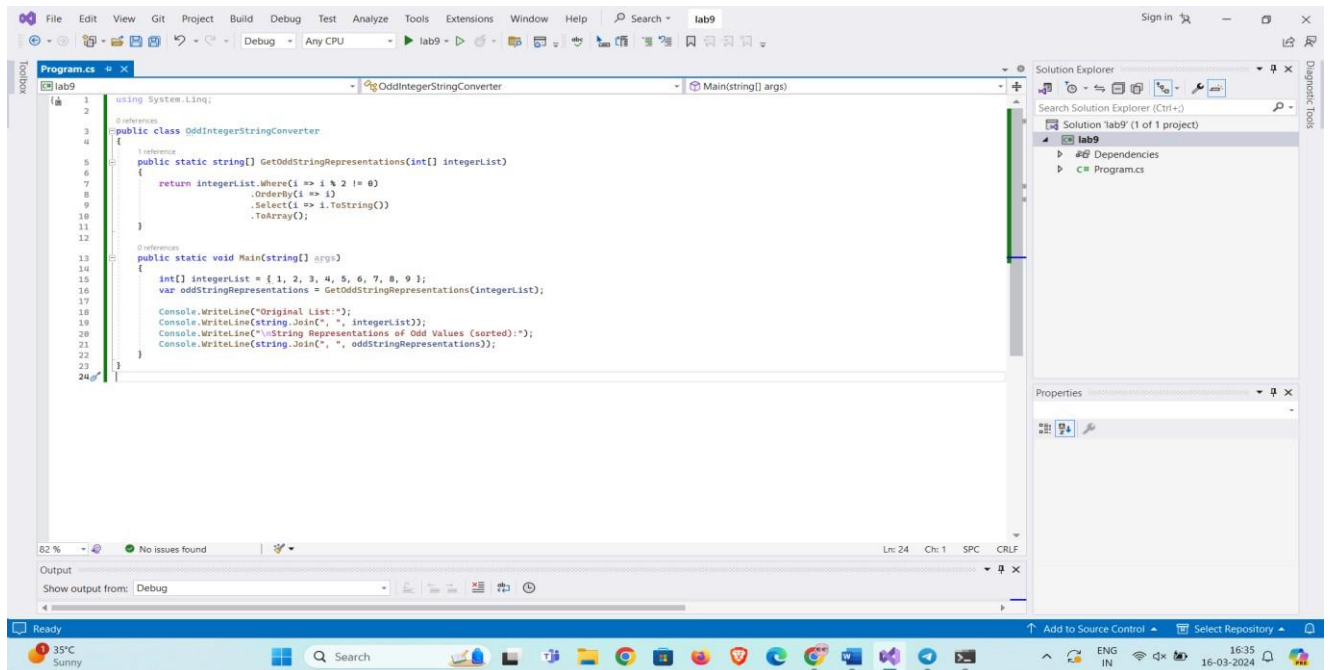
TASK 4: A positive integer **K** and a sequence of non-empty strings **stringList** are given. Strings of the sequence contain only numbers and capital letters of the Latin alphabet. Get from **stringList** all strings of length **K** ending in a digit and sort them in ascending order.



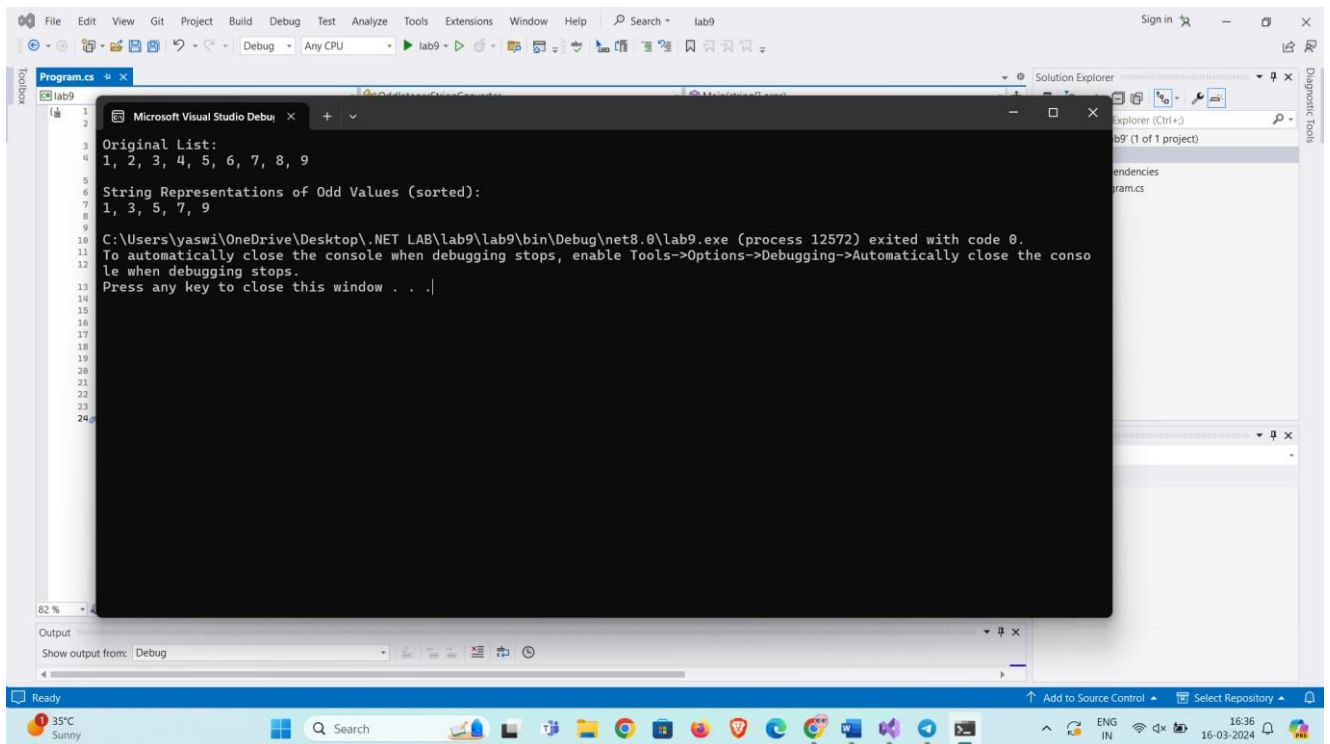
OUTPUT:



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.



OUTPUT:



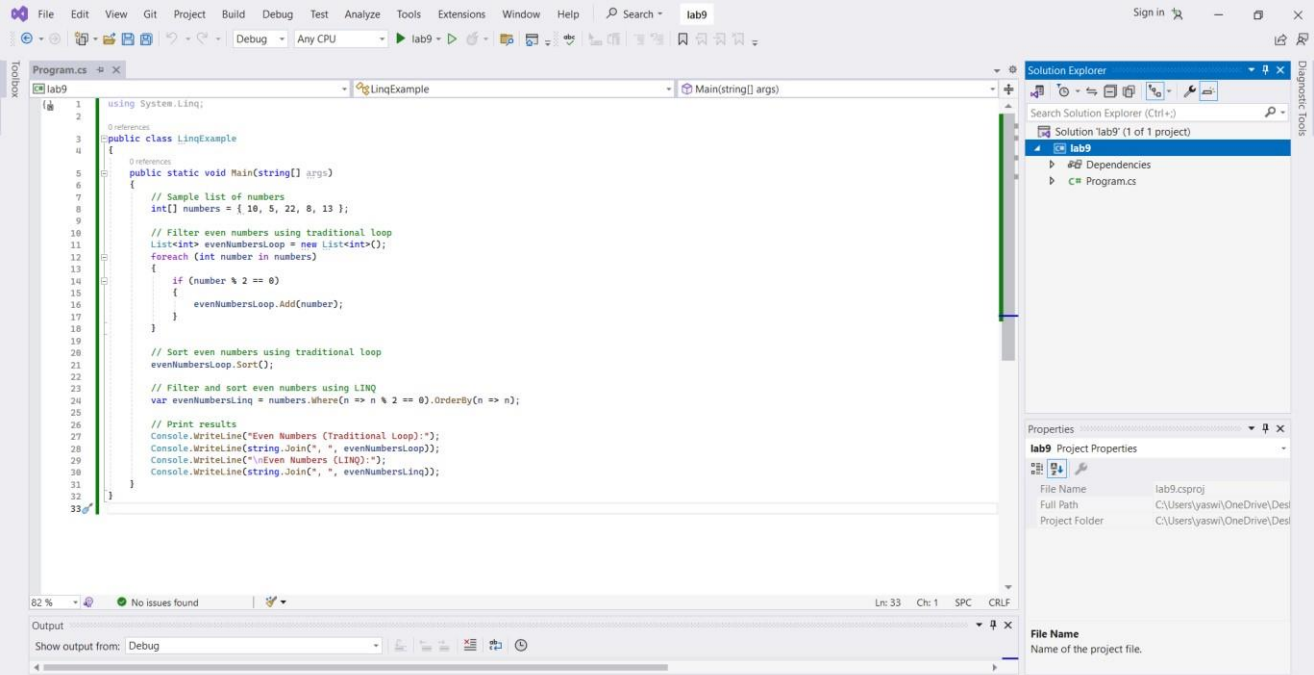
POST-LAB

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.



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

```
1 using System.Linq;
2
3 public class LinqExample
4 {
5     public static void Main(string[] args)
6     {
7         // Sample list of numbers
8         int[] numbers = { 10, 5, 22, 8, 13 };
9
10        // Filter even numbers using traditional loop
11        List<int> evenNumbersLoop = new List<int>();
12        foreach (int number in numbers)
13        {
14            if (number % 2 == 0)
15            {
16                evenNumbersLoop.Add(number);
17            }
18        }
19
20        // Sort even numbers using traditional loop
21        evenNumbersLoop.Sort();
22
23        // Filter and sort even numbers using LINQ
24        var evenNumbersLinq = numbers.Where(n => n % 2 == 0).OrderBy(n => n);
25
26        // Print results
27        Console.WriteLine("Even Numbers (Traditional Loop):");
28        Console.WriteLine(string.Join(", ", evenNumbersLoop));
29        Console.WriteLine("Even Numbers (LINQ):");
30        Console.WriteLine(string.Join(", ", evenNumbersLinq));
31    }
32 }
33
```

The Solution Explorer on the right shows the project structure with 'lab9' and 'Program.cs'. The Properties window shows the project properties for 'lab9'. The Output window at the bottom shows the results of the program execution.

Output:

```
Even Numbers (Traditional Loop):
10, 22, 8
Even Numbers (LINQ):
10, 22, 8
```

OUTPUT:

