**Longest Common Substring**

Mohammed Towseef

Advanced Algorithms

Concordia

Farah Kamw

July 21, 2024

**Documentation: Longest Common Substring.**

**Problem Analysis**

The assignment was on the usage of C++ language in solving the Longest Common Substring problem. In this problem, one must determine which substring is the longest and occurs in both of the given strings. The task includes generating and storing a table containing the longest suffixes' lengths shared between the substrings. The table is dynamically generated, and last but not least, extracting the longest suffix.

**Approach**

Besides, the solution implements dynamic programming to solve the problem by minimizing its complexity. A 2D table records the lengths of maximal suffixal matches of substrings. By comparing individual characters of both strings, this table is filled, and thus, the longest common substring is found.

**Steps**

**1. Define the Longest Common Substring Function:**

This function controls the 2D table and the procedure of filling this table with the help of comparing characters from both the strings.

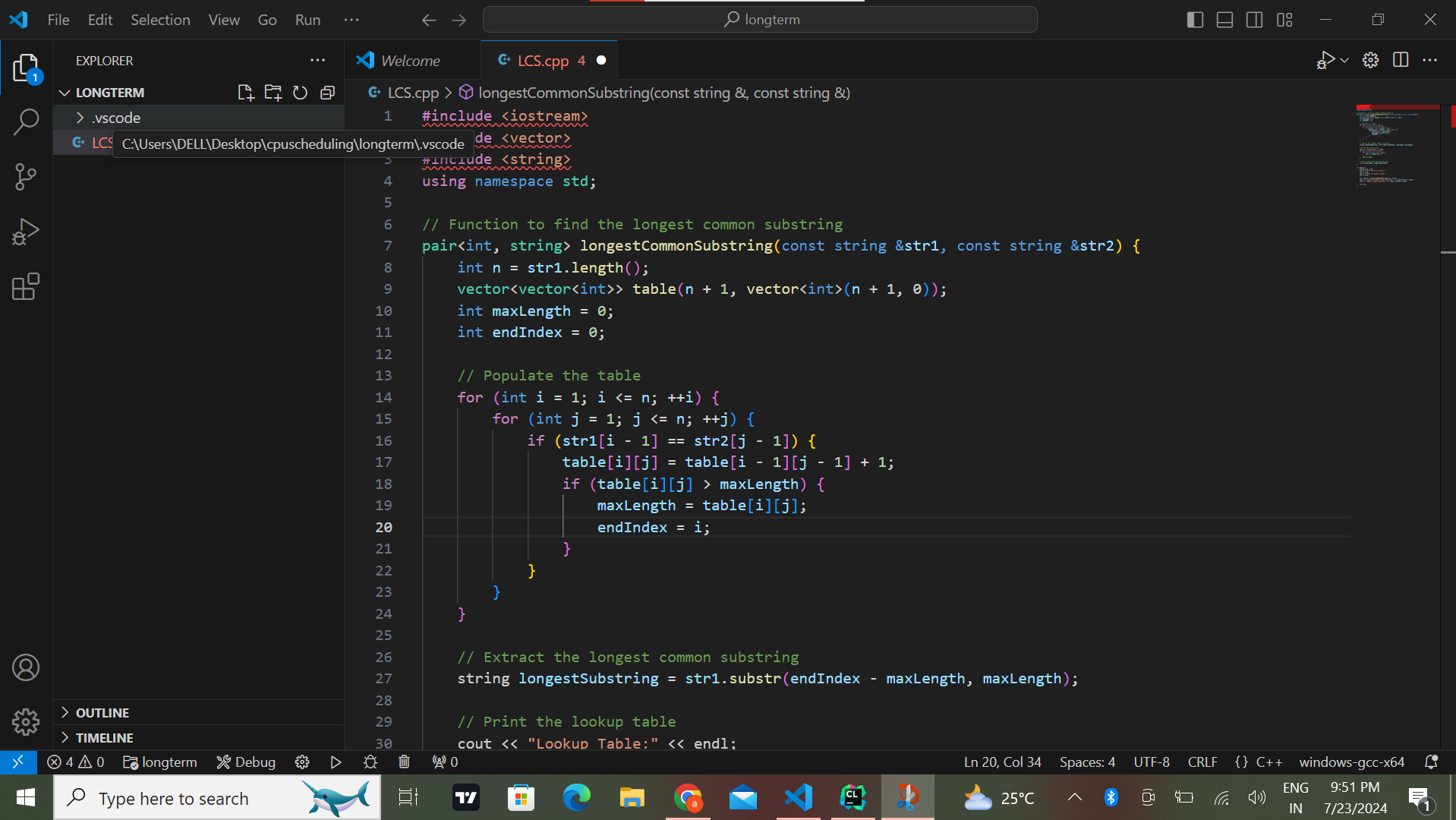
**2. Implement Methods:**

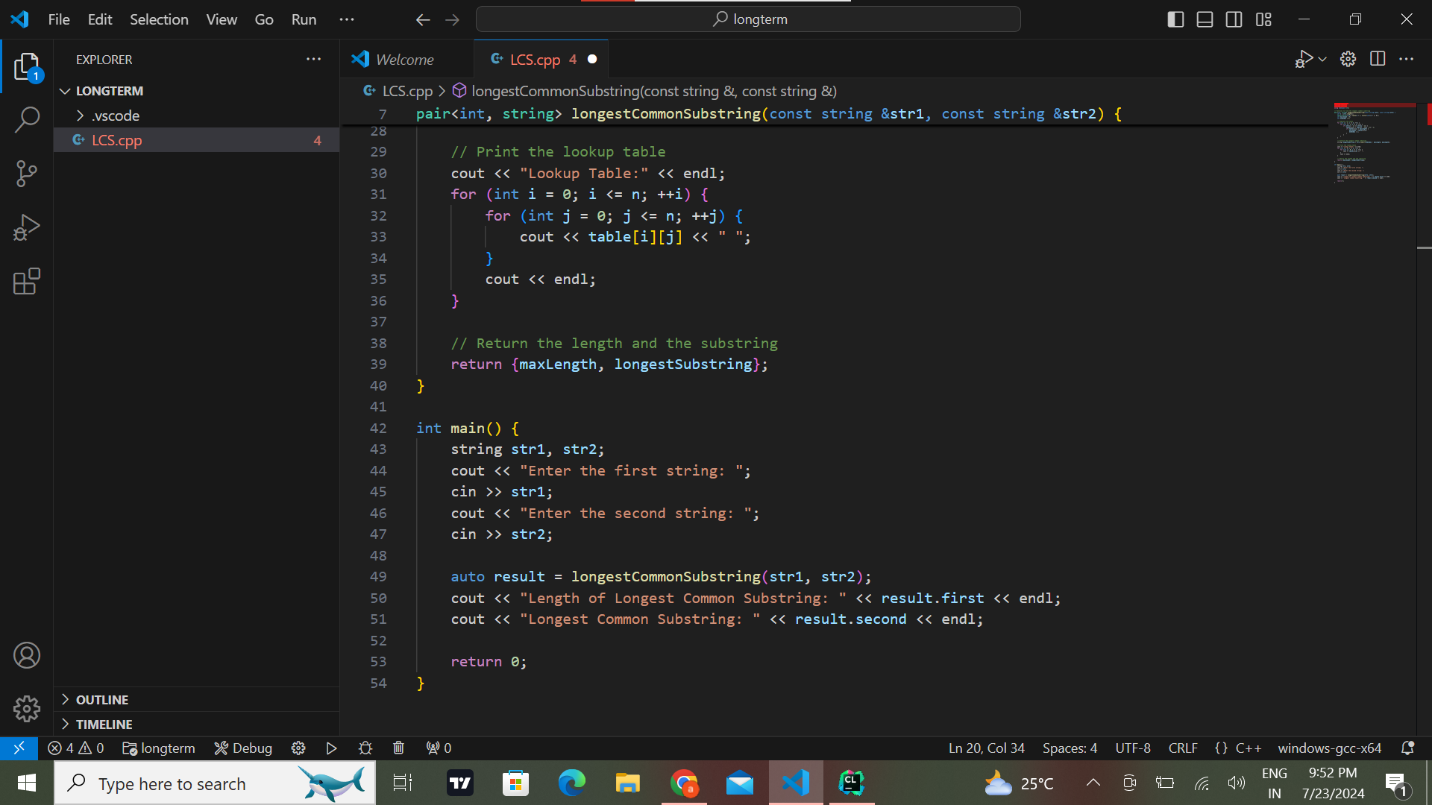
**- `addElement(int row, int col, int value)`** does not apply here, but the manner used to fill this table is not different from the above-given examples.

**- `printSparseMatrix()`:** We have already filled in the 2D table with the lengths of suffixes that are common for the given strings.

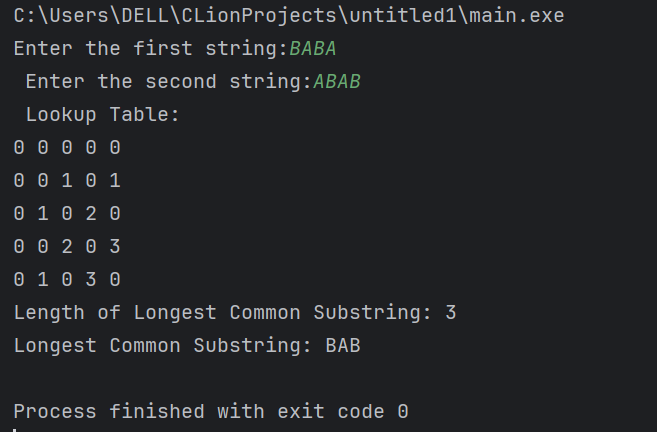
**- `printMatrix()`:** This is similar to reconstructing and printing the longest common substring based on the filled-in table.

**Code Explanation:**

****



**OUTPUT :**



**Student Reflection on Problem-Solving**

**How much time did this assignment take from you in total?**

This task took me about 2-3 hours to accomplish, starting with gaining awareness of the given problem, developing the corresponding solution, and checking it against different inputs.

**Based on the effort you could have put in, what letter grade do you think your work deserves?**

Looking at my effort in the program, I believe I deserve an A on this assignment. While implementing the solution, I strictly followed the assignment instructions; I implemented the solution for this program using dynamic programming.

**If you have to grade yourself according to your proposed solution, what grade rate would you give for this project?**

Thus, I can conclude that considering my final solution, I should better get an A grade. The program executes the targeted objective of finding the longest common substring, manages user input, and generates the lookup table, as expected.

Concisely state all that doesn’t work in your solution, explain how you tried to tackle the problem, and where you think you lacked. Conversely, nothing about the solution expressed in the play doesn’t functions as designed, especially for the two main female characters, Diana and Paulina. Nonetheless, one of the repetitive issues I faced at the start of the task was the challenge of formatting the lookup table to be properly viewable with easy readability. For this, I concentrated on the right traversal of the 2D vector and how exactly to display the vector. Also, I made sure that while organizing the strings with multiple common substrings of the same length, the program provided one valid result.

**How I Tried to Solve the Problem:**

**1. Understanding the Problem:**

Before writing the solution, I reviewed the problem statement to analyze all the specifications. Thus, I could represent the goal of the longest common substring and update the lookup table in terms of dynamic programming.

**2. Designing the Approach:**

I decided to use dynamic programming because it optimally solves problems by dividing problems into sub-problems and using a two-dimensional table to store the results of sub-problems.

**3. Writing the Code:**

I coded the logic within the `Longest Common Substring’ function. What I did was making sure that the table provided was filled in properly and making sure the longest common substring was extracted using the maximum value from the table.

**4. Testing the Solution:**

Testing the solution with various inputs, beginning with the given examples and expanding by me, ensured the correctness of the code and the possibility of its optimization. To do this, I ensured that I designed a program that could easily handle user inputs and produce the right outputs based on the set format.

**Where I Struggled:**

**1. Handling Edge Cases:**

At first, I had issues when trying to think through, or to experiment with, different test cases, like strings that contain two or more different LSS of the same maximal length. Something more severe in terms of scale and implications made it necessary to be more attentive to such situations;

**2. Ensuring Dynamism:**

Another fine thread to consider was accepting the users' inputs to change the code dynamically while ensuring the code remained correct. I had to define good input specifications and a naturalistic user interface.

**3. Debugging:**

Debugging was a complex process owing to validation of the solution obtained was right for all possible test cases. The code needed to be tested to guarantee that the table was populated correctly with the string in the input line and the correct substring was used.

**Conclusion:**

As for me, I think my solution is good for the stated problem and solves it in reasonable time, taking less resources. The program certainly provides flexibility and extensibility possibilities, but handling the constraints, notably the inputs and possible positions and manipulations of the lists, can be more fine-tuned. This assignment was a good way of revising the dynamic programming course and renewed my practice in efficient problem-solving in C++. It also pointed out areas that needed further development in the future, which would indeed help the training of a programmer.