CSCI 335 **Fall 2022**

ExtraCredit for Course Grade

4 points (2 points for each question)

Due noon December 19, 2022

This assignment has just this one deadline - no late submissions past this deadline will be accepted.

This is an individual assignment

Instructions:

1. **Read and follow the contents of 335 Fall22 335 Programming Rules document on the blackboard (Course Information Section).**
2. **Read the question description below.**
3. **Submit only the files requested in the deliverables at the bottom of this description to Gradescope by the deadline.**
4. **Each question is worth 2 course grade points. The autograder will be able to assign full credit for each question. However manual checks will be done and the README reviewed and if specific requirements are not met, the grade will be lowered with explanation.**
5. **Since this is an extra credit question, tutors will not be able to help with the questions directly. They can however help explain concepts.**

Q1: (2 EC points for course grade) - Longest Common Subsequence

Attachments provided: subsequence.cc

For this question, there are 1 visible test and 1 invisible/hidden test.

Subsequences are sequences within a string that appear in the same relative order but are not necessarily contiguous. For example, if given a string "123456", then some potential subsequences are, "123", "12", "456", "14", "236", "46", etc.

The longest common subsequence problem is as follows:

Given two sequences **A= $a_1,a_2,\ldots a_m$** and **B= $b_1,b_2,\ldots b_N$** find the length, **k**, of the longest subsequence **C= $c_1,c_2,\ldots c_m$** such that **C** is a subsequence (not necessarily contiguous) of both **A** and **B.**

Example 1: If **A = dynamic** and **B = programming** then the longest common subsequence is **ami** and has a length of **3**.

Example 2: If **A = apples** and **B = oranges** then the longest common subsequence is **aes** and has a length of **3**.

Write an algorithm to solve the longest common subsequence problem in O(MN) time.

Your program should take two arguments, word_a and word_b.

Given a call, **./subsequence <word_a> <word_b>**, your program should output in the following format.

<length_of_longest_subsequence>
<longest_subsequence>

Example: Given, **./subsequence apples oranges** your program should return:

3
aes

Q1 Deliverables:

- subsequence.cc (modified)
  - subsequence_driver()
- README file (one comprehensive README about Q1 and Q2)

Q2: (2 EC points for course grade) – Optimal Matrix Multiplication

Attachments provided: optimal_multiplications.cc, optimal_ordering.cc, dimension_file.txt, dimension_file2.txt

For this question, there are 1 visible test and 1 invisible/hidden test.

Given the sizes of several matrices, calculate the optimal multiplication ordering using dynamic programming. The sizes will be presented in a file containing dimensions in a sequence:

For example, dimensions_file.txt can be

50
10
40
30
5

That means that c0 = 50, c1 = 10, c2 = 40, c3 = 30, and c4 = 5. Therefore. the matrices to be multiplied have sizes:

Matrix 1: 50 x 10
Matrix 2: 10 x 40
Matrix 3: 40 x 30
Matrix 4: 30 x 5

Obviously when the dimensions_file.txt contains N numbers, then the matrices to be multiplied are N – 1.
Write a program that runs as follows:

./optimal_multiplications <dimensions_file>

The program should produce the optimal number of multiplications.

./optimal_multiplications dimensions_file.txt, should output a single number.

10500

Note: This output is not necessarily representative of any particular input.

Q2 deliverables:

- optimal_multiplications.cc (modified)
  - multiplication_driver()
- README file. (one comprehensive README for Q1 and Q1)