CSCI 335 **Spring 2020**

Final Exam: Section 1 Programming Questions[1]

50 points (45 points deliverables, 5 points for design) +10 points ExtraCredit

Due 11pm May 15th, 2020

No extensions will be given to the submission deadline. Plan to submit earlier than 11pm of the deadline to avoid last minute issues.

READ THESE INSTRUCTIONS CAREFULLY BEFORE EXAM!

1. The final exam has two sections – a programming section and written section.
2. This is an open book, open notes take-home exam with a 1 week deadline.
3. Read all questions before you start the exam and plan your time well. Read midterm exam discussion powerpoint on blackboard.
4. This is the programming section. It should be an individual submission.
5. You are not permitted to discuss any part of this exam with anyone other than 335 course staff. You can ask only questions to the course staff via blackboard or email. Final exam FAQs, one each for the written and programming sections, will be maintained.
6. Submit only the files requested in the deliverables at the bottom of this description to appropriate Gradescope link by the deadline.
7. Only clarification of question statements and possible issues with gradescope script, if any, will be addressed. No code debugging or conceptual answers will be provided by staff- this is an exam. **READ ALL PROBLEM REQUIREMENTS CAREFULLY.**
8. All work submitted must be your work only. If you refer to a source other than the text book, state that in your answer. Reread Academic integrity requirements for the class in class syllabus and programming documents. We will strictly enforce this policy for this take-home final. Code comparison software will be used. Clear violations including contract cheating using StackOverflow, Chegg etc. will lead to automatic zeros.
9. It is your responsibility to make your answers readable. If we can't read /find it, the answer will get 0 points.
10. Submit a README file for each question.

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

[1] This document or its contents should not be shared unless approved by the instructor.

Programming Q1: (20 points) - Longest Common Subsequence
Attachments provided: subsequence.cc

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 all Q's)`

Programming Q2: (30 points) – Optimal Matrix Multiplication

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

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 about all Q's)`

Programming Q2 Extra Credit (10 points) – Optimal Matrix Multiplication Ordering

For 10 points of extra credit, submit **`optimal_ordering.cc`**, that upon the input: **`./optimal_ordering <dimensions_filename>`** outputs the correct optimal matrix multiplication order.

You should use parentheses, and the matrices should be named A through Z, in order, left to right. You can and should use your work in Q2.

For example:

`./optimal_ordering dimensions_file.txt` should print:

```
(A(B(CD)))
```

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

More potential output examples: (not necessarily representative of any files)

```
((AB)((CD)E))
((A(BC))D)
```

Your output should be of this form, with parentheses surrounding every grouping, including the outermost. You can assume that there will not be more than 26 matrices in a given dimensions file.

Please make sure that all functionality is called from the function **`ordering_driver().`**

Q2 EC deliverables

- `optimal_ordering.cc (modified)`
  - `ordering_driver()`
- Readme file. `(one comprehensive README about all Q's)`