CSC505 Jennings
Homework 5, Spring 2020
Please implement the programs and answer the 9 questions.

This assignment may be done alone or in groups of 2 or 3 people. Turn in one copy of the written part of the assignment on Gradescope, and enter the names of each team member during the submission process.

I. Overview:

Do not discuss this homework with other teams.

In this assignment, you will implement a version of the popular `diff` program using (1) a string hash function and (2) an algorithm for the Longest Common Subsequence problem, using the Dynamic Programming paradigm.

For this assignment, you can turn in a PDF in almost any format – there is no template to follow. On Gradescope, you will have to indicate where in your PDF we will find the answer to each question.

You must provide (via NCSU github) to the teaching staff any source code you use and scripts that can be used to build any code and run any/all steps needed to reproduce your results.

And, as always, you must cite the origin of all source code, which may be original to your team or individual team members, or may be appropriately obtained from elsewhere.

These programming languages are acceptable for source code used in this assignment:

Java 12
Python 3.7 or 3.8
C (C11 or ANSI)
C++ 17

Checklist:

- ☐ Read and answer the questions on the following pages.
- ☐ Implement from scratch OR use code found online or in a library (which you must cite) the algorithms needed for this assignment.
- ☐ Tell Gradescope who worked on your team assignment. (One, two, or three people may work on it.)
- ☐ Do not forget to cite all resources you used, including our textbook.
- ☐ Add the teaching staff to your NCSU github repository.
- ☐ Be sure to include a link to your NCSU github repository.

II. Implementation

A. Implement a hash function for strings. Use a *modulo* function to compress the resulting value to 10 bits, i.e. to be in the range [0, 1023].
For example:

```
hash10("Hello!") ➔ 429
hash10("") ➔ 0
hash10("\n\n\n") ➔ 74
```

Your compressed hash values may differ.

B. Use your hash function to implement a `hashfile` program that takes a filename on the command line and produces a 10-bit hash of each line.
For example:

```
$ hashfile f1.txt
517
567
263
$ hashfile f2.txt
517
263
76
765
```

You can see that f1.txt has 3 lines, and f2.txt has 5 lines. The hash values printed by your program will probably be different.

If your program requires an interpreter, like python3, then make a bash script that runs your program using the interpreter. The teaching staff must be able to invoke your program as shown.

C. Implement the dynamic programming algorithm for `diff`, using the dynamic programming approach to the Longest Common Subsequence problem. Use the code from your `hashfile` implementation. (You do not need to run `hashfile` and use its output – just be sure to re-use your hashing and file i/o code.) The output of `diff` should be two lists of line numbers, one for file1 and one for file2. The lists are the same length and contain the line numbers *(with 1 as the lowest line number)* that are the same in the two files.

For example, we can see by the hash numbers in the previous example that line 1 is the same in f1.txt and f2.txt (they hash to 517); and line 3 in f1.txt is the same as line 2 of f2.txt (they hash to 263). All other lines are different.

Sample output from `diff`:

```
$ diff f1.txt f2.txt
1 2
1 3
```

You can check your program using the sample files provided at https://github.ncsu.edu/jajenni3/csc505/tree/master/HW5.

D. FOR BONUS POINTS:
Implement a program `diffprint` that uses the code you wrote for `diff` and prints a simplified form of the "standard" diff output showing the contents of both files. Lines that are common to both files are printed with two spaces in front of them. Lines present only in the second file are printed with "+ " (plus, then space) in front, and lines present only in the first file are printed with "- " (minus, then space) in front.

For example:

```
$ diffprint f1.txt f2.txt
  Hello
- there
  world.
+ Really!
+ Yep.
```

III. Questions:

1. Names of the people in the team:
   (without this, you will receive 0 for the assignment)

   _____

   _____

   _____

2. NCSU github URL: (without this, you will receive 0 for the assignment)

   _____

3. Citations for code: (without this, you will receive 0 for the assignment)
   Please cite non-code resources in your answers to specific questions.

   _____ (Use more space as needed.)

   _____

   _____

4. What is the output of your `diff` program on files ex41.txt, ex42.txt?

Show the command line invocation and the output

5. What is the output of your `diff` program on files ex42.txt, ex41.txt?

Show the command line invocation and the output

6. If you implemented `diffprint`, give the output here… Else leave blank
   a. For files ex61.txt, ex62.txt
   b. For files ex62.txt, ex61.txt
   c. For files ex66a.txt, ex66b.txt

Show the command line invocation and the output

7. Your implementation of LCS requires how much space? Your answer should use big-Theta, because your algorithm cannot use less and will never need more.

A short answer is appropriate. Be sure to explain the meaning of any variables like m or n.

8. How do you know that the 10-bit hash code required by this assignment is sufficient to allow your `diff` program to work correctly? (How would you know if it was not sufficient?)

A few sentences should suffice.

9. Suppose you were tasked with writing a `diff` program that would work reliably on input files that were long (up to, e.g. 10,000 lines) and could have long lines (up to, e.g. 5,000 bytes).

   a. What are the implications for the choice of hash function? (Describe what properties you would want in a hash function.)

There are several points you could potentially make here.

   b. What are the implications for actual memory usage? I.e. calculate a reasonable approximation of how much memory your program might need in the worst case.

First identify the main consumer(s) of memory that is needed during processing of LCS.