This assignment may be done 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.

CSC505 Jennings
Homework 3, Spring 2020
Please read, implement, and answer all 4 sections below.

I. Names of team members:

____

____

____

II. Overview:

In this assignment, you will design algorithms for solving string-related problems. You will implement some of your designs, as indicated in the next section. (The Analysis Questions section prompts further design activity.)

You may also need to write some code (or use existing programs) to analyze the provided sample data.

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 any source code used in this assignment:

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

III. Implementation tasks:

Using the data files in the HW3 directory of https://github.ncsu.edu/jajenni3/csc505 as sample input files, design and implement solutions for the following problems. Your solution should have a command line interface so that automated tests may be easily scripted.

DIALOG: Given a list of input files similar to the sample files (but not previously known), extract all the dialog into a separate output file. In the output, retain the quotation or other marks that separate the dialog from the narrative.

DIALOG SEARCH: Given one input file similar to the sample files (but not previously known), and a search string, determine if the search string appears within dialog in the sample file. If it does, then output each Chapter number/title (equivalently for anthologies, the number/name of the individual work) in which the search string appears.

You may require additional parameters to be supplied on the command line, or from a configuration file. In that case, document what each one does, and provide examples of their usage.

Put your code and data into a GitHub repository on github.ncsu.edu in a branch called "HW3" (upper case). Give read access to the teaching staff (3 TAs and Dr Jennings – email addresses on Piazza).

IV. Analysis questions:

1. Give the URL to an NCSU github repository containing your code, scripts, and data.

2. Provide citations for your source code and any data processing programs used. For command line utilities like grep, indicate the version (e.g. using grep --version). For any installed software, like a Python package, give the version and the command needed to install it.

3. Demonstrate that your implementations work by providing a script for the DIALOG problem and another script for the DIALOG SEARCH problem.
   a. The script for DIALOG should run your program on both sample input files, producing output files that you should commit to your repository.
   b. The script for DIALOG SEARCH should run your program multiple times, on a variety of search strings, using each of the sample files.

4. Documenting your approach to DIALOG and to DIALOG SEARCH. I.e. explain in words (and/or diagrams) how your algorithm finds the needed pieces of text. Importantly, this explanation documents (1) all assumptions you make about the input data, and (2) any additional parameters needed for your programs to run.

5. How can a user determine whether a new input file (never seen before) will be acceptable to your programs? If you used any additional parameters, how does the user determine what values to provide?

6. Consider the two sample input files, and the output of your DIALOG solution. How did you verify that your implementation works correctly?

7. Consider the two sample input files, and the test cases you scripted for DIALOG SEARCH. How did you verify that your implementation works correctly?

8. What is the worst-case asymptotic complexity of your DIALOG solution as a function of the m characters of total input? And for DIALOG SEARCH, as a function of m characters of input text and an n-character search string?

9. DIALOG design question: How would you change your solution to DIALOG if you expected the same input file to be given multiple times over the course of days, weeks, or even years of people using your program? Consider how you would identify the input file, and the fact that persistent storage is always limited. Give the benefit and the cost of your approach.

10. DIALOG SEARCH design question: How would you change your solution to DIALOG SEARCH if you knew in advance the collection of files (the “corpus”) that your program would have to search? Consider how the user would identify which file they wanted to search. How could additional files be added to the corpus? Give the benefits and costs of your approach.