```
/* CMPUT 201 Assignments                                                */
/* Dues:                                                                */
/* #1:    11:55pm, September 29, 2019;                                  */
/* #2:    11:55pm, November 3, 2019;                                    */
/* #3:    11:55pm, December 6, 2019                                     */
```

(Mandatory assignment cover-sheet; without it, your work will not be marked.)

Submitting student: Nawalage Dinuka Ravindu Cooray

Collaborating classmates: Mentioned in the source code

Other collaborators: Mentioned in the source code

References (excluding textbook and lecture slides): Mentioned in the source code

Regardless of the collaboration method allowed, you must always properly acknowledge the sources you used and people you worked with. Your professor reserves the right to give you an exam (oral, written, or both) to determine the degree that you participated in the making of the deliverable, and how well you understand what was submitted. For example, you may be asked to explain any solution that was submitted and why you choose to write it that way. This may impact the mark that you receive for the deliverable.

So, whenever you submit a deliverable, especially if you collaborate, you should be prepared for an individual inspection/walkthrough in which you explain what every line of assignment does and why you choose to write it that way.

From Assignment #1, you have learnt what a sequence is, as well as a subsequence, a common subsequence, and a longest common subsequence (LCS for short).

Let T be a sequence of an even length $2k$, for some non-negative integer k. If its prefix of k symbols (or, the first k symbols in T) are the same as its suffix of k symbols (respectively, the last k symbols in T), then T is called a *tandem* sequence. For example, $T[8] = 31103110$ is a tandem sequence.

Given a sequence S, if T is a subsequence of S and T is tandem, then T is called a *tandem subsequence* of S. One can then similarly define what a *longest tandem subsequence* (LTS for short) of S is.

Let R be a sequence. The *inverse* sequence of R is the sequence by reading the symbols in R backwards. For example, the inverse of 0113 is 3110. If the inverse of R is identical to R, then R is called a *palindrome* sequence. For example, $R[9] = 011303110$ is a palindrome sequence.

Given a sequence S, if R is a subsequence of S and R is palindrome, then R is called a *palindrome subsequence* of S. One can then similarly define what a *longest palindrome subsequence* (LPS for short) of S is.

Assignment #2 is on computing an LTS and an LPS for any given sequence of length up to $10,000$ over the digit alphabet $\Sigma = \{0, 1, 2, \ldots, 9\}$. Recall that your C program for Assignment #1 is able to read in a sequence over Σ of length up to $1,000$, and now you need to increase the capacity to $10,000$. (If you didn't do Assignment #1, you might consider purchasing a solution by paying 2 marks from your overall total.)

The following list contains the specifications for Assignment #2 (10 marks in total):

1. Write a single-file multi-function C program with multiple functionalities (i.e. objectives). Your program should succeed in the first three functionalities specified in Assignment #1, with an increased capacity to read in two sequences of length up to $10,000$ and print them out (functionality 1).

   Let us change slightly the front interface as follows (with an example input from the user):

   ```
   Enter the first sequence: 31102830253182928l8318143
   Enter the second sequence: 1311230172803163125l841324
   ```

2. Your program computes an LTS for the first given sequence and prints it out to the `stdout` (functionality 2, the following `1283312833` is indeed an LTS for the first sequence above).

   ```
   # an LTS (length = 10) for the first sequence is:
   1283312833
   ```

3. Your program computes an LPS for the second given sequence and prints it out to the `stdout` (functionality 3, the following `3112136312113` is indeed an LPS for the second sequence above).

   ```
   # an LPS (length = 13) for the second sequence is:
   3112136312113
   ```

4. (Coding style requirement:) You should partition your source code into multiple functions, with the computation of an LTS and the computation of an LPS in two separate functions; if your code still computes an LCS, then the computation should be in a separate function too.

   //End of description of Assignment #2.