# CSE 5311-001 Lab Assignment 1

Due March 7

**Goals:**

1. Review of dynamic programming as applied to the longest strictly increasing subsequence (LSIS) problem.

2. Review of elementary longest common subsequences.

3. Preview of Notes 14 and approaches to longest common subsequences.

**Requirements:**

1. Write a C program to compute the longest common subsequence of two sequences in two different ways:

    a. The elementary cost matrix method using $\Theta(mn)$ space.

    b. The LSIS method intended for use in sparse situations with relatively large "alphabets":

        1. For each of the 256 alphabet symbols, determine the positions (*descending* order) where the symbol appears in the second sequence. *Do not do 256 passes over the second sequence!* (Think about counting sort . . .)
        2. Produce an intermediate sequence by *replacing* each symbol in the first sequence by its positions from the second sequence.
        3. Compute a LSIS of the intermediate sequence.
        4. The sequence of values from the LSIS may be used as indexes to the second sequence to obtain an LCS.

    c. In all situations, the solution you find for both methods should be *identical*.

2. The input for the two input sequences will be formatted as:

    a. The first line of the input will be two values, `m` and `n`, giving the lengths of the two input sequences. These will not exceed `25000`.
    b. The next `m` lines will each contain a single integer in the range `0 . . . 255`.
    c. A line with the value `-1`.
    d. The next `n` lines will each contain a single integer in the range `0 . . . 255`.
    e. A line with the value `-1`.

3. The output from your program should go to standard output, not a file:

    a. The first line should be the length of the LCS found by both methods.
    b. Each of the remaining lines should have one element of the LCS found by both methods.
    c. A single line with the value `-1`.
    d. Before emitting this output, your program should explictly check that the two methods (independently) produced the same result. *Do not print a result if they differ!*
    e. The last line of your output should provide the CPU time for each method.

4. Submit your C code on Canvas before 3:45 p.m. on Monday, March 7. Your program must compile and execute on `omega.uta.edu`.

**Getting Started:**

1. You may use publicly-available code (or library routines) to get started, but give appropriate credit.

2. Keeping memory usage low is important.

Sample Input:

```
9 9
7
8
9
8
7
7
8
9
-1
7
7
8
6
9
8
9
```

```
First Sequence
0 1 2 3 4 5 6 7 8
7 8 9 9 8 7 7 8 9

Second Sequence
0 1 2 3 4 5 6 7 8
7 7 8 8 6 9 8 7 9

Positions for Symbols in Second Sequence
6:  4
7:  7 1 0
8:  6 3 2
9:  8 5

Replacing Symbols in First Seq. by Positions of Second Seq.

0     1     2     3     4     5     6     7     8
7     8     9     9     8     7     7     8     9
7 1 0 6 3 2 8 5   8 5   6 3 2 7 1 0 7 1 0 6 3 2 8 5

Longest Strictly Increasing Subsequence
0 2 5 6 7 8
-1
7 8 9 8 7 9 Longest Common Subsequence
```

Sample Output:

```
6
7
8
9
8
7
9
-1
```

Matrix Version:

```
789987789
778869879
LCS is 789879, length==6
      7  7  8  8  6  9  8  7  9
   0  0  0  0  0  0  0  0  0  0
7  0  1  1  1  1  1  1  1  1  1
8  0  1  1  2  2  2  2  2  2  2
9  0  1  1  2  2  2  3  3  3  3
9  0  1  1  2  2  2  3  3  3  4
8  0  1  1  2  3  3  3  4  4  4
7  0  1  2  2  3  3  3  4  5  5
7  0  1  2  2  3  3  3  4  5  5
8  0  1  2  3  3  3  3  4  5  5
9  0  1  2  3  3  3  4  4  5  6
```