**CSCE A412: Evolutionary Computing**

**Project 1: Genetic Algorithms**

**Due Wednesday, February 15, 2017, at 11:59 PM**

Wikipedia describes the Longest Common Subsequence problem as follows:

The **longest common subsequence** (**LCS**) **problem** is the problem of finding the longest subsequence common to all sequences in a set of sequences (often just two sequences). It differs from problems of finding common substrings: unlike substrings, subsequences are not required to occupy consecutive positions within the original sequences.

For example, the LCS of the sequences “president” and “providence” is “priden”.

For this assignment, your task is to write a genetic algorithm (GA) that solves the LCS problem for two interactively specified sequences of characters. Here are some pointers:

1. The LCS will never be longer than *k*, where *k* is the length of the shorter of the two input sequences.
2. Each of the candidate solutions in your random initial population will be a string of length *k* consisting of 1s and 0s indicating whether or not the corresponding character from the shorter sequence is included in the subsequence. For the example shown above, *k* = 9 and candidate solutions “011011011” and “100000101” represent subsequences “reidnt” and “pet”, respectively, while the optimal solution (“priden”) would be represented as “110011110”.
3. The hardest part of solving any non-trivial problem with a GA is to develop a good, consistent fitness function. Clearly, you’ll want to strongly reward candidate solutions representing subsequences that appear entirely in the longer sequence (e.g., “id”); you may also want to reward longer sequences (i.e., those with more 1s), or heavily penalize a subsequence not found in the longer sequence (e.g., “st”).
4. I recommend using standard fitness-proportionate (“roulette wheel”) selection to create your mating pool, but you may also consider tournament selection.
5. Once the mating pool has been formed, use standard one-point crossover and bit-wise mutation to create the next generation of candidate solutions. The probability of crossover should be high (e.g., 95%), while the probability of reproduction and mutation should be relatively low (e.g., 5% and 1/*k*, respectively).
6. Debug your software using relatively simple example problems like the one shown above. Once your program is ready, test it on at least five non-trivial problems (for example, on input strings containing dozens or even hundreds of characters; you may want to hard-code longer strings to avoid having to type them in interactively!). For each test run, record the control parameters used (M, G, pc, pm, pr), the two input strings, and the resulting LCS, and the number of generations it actually took to find the LCS. (NOTE: there may be more than one optimal solution; your GA should record the number of generations it needed to find the *first* LCS.)

As with any programming project, your program should exemplify principles of high-quality computer software design. Program-level documentation should indicate the name and preferred email address of each author.

Each team should also write a brief report that includes each of the following items:

1. The name and preferred email address of each team member.
2. A table representing the control parameters (M, G, pc, pm, pr), the types of crossover and mutation used, the inputs strings used, and the results (i.e., the best-of-run LCS and the number of generations needed to find it) from each of the five non-trivial test runs.
3. A brief summary of the results of this programming assignment. Comment upon the quality of your results: did your GA always find the LCS? If so, how many generations did it take? If not, why not? What do you think made some LCS problems harder than others (i.e., took more generations to solve)? Did your GA scale well, i.e., how did it perform for longer strings?

Submit your report and software project (including source code and executable) via email. I will read your report, examine your source code, run your program, and return commented and graded reports to you via email.