Algorithms

1. Let S_1 and S_2 be two strings of length m and n respectively. We say that a string S_3 is obtained by interweaving S_1 and S_2 if S_3 can be obtained by inserting **all** characters of S_2 into S_1 without disturbing the relative ordering of the characters within S_1 or S_2. For example, if $S_1 = bad$ and $S_2 = doggy$ then *baddoggy*, *doggybad*, *dobadggy*, *bdaodggy* are all obtained by interweaving S_1 and S_2 but *bdadgogy*, *baddog* are not. Denote by $interweave(S_1, S_2)$ the set of all strings that can be obtained by interweaving S_1 and S_2. Give pseudocode for a simple algorithm that given two strings S_1 and S_2 prints all different strings in $interweave(S_1, S_2)$ exactly once. What is the worst-case running time of your algorithm (as a function of m and n)?

   30 pts

2. Given two strings S_1 and S_2 of length m and n respectively, and a third string T of length t, we want to determine if **any** string from $interweave(S_1, S_2)$ occurs (as a substring) in T. For example, if $S_1 = bad$, $S_2 = doggy$ and $T = cbbdobadggycbb$ then the answer is "yes" because the string *dobadggy* occurs as a substring in T starting at position 4. Give a polynomial time algorithm to solve this problem. Analyze the space and time complexity of your algorithm.

   70 pts

   Hint: The problem is similar to the longest common subsequence problem. So, think recursively and use dynamic programming. In particular, it might help to think recursively about the function $M(i, j, k)$, where $M(i, j, k) = TRUE$ iff $S_1[1 \dots i]$ and $S_2[1 \dots j]$ occur interleaved at the very end of string $T[1 \dots k]$.