Problem 1. (20 = 4 + 8 + 8 points). $C$ is a substring of a string $A$ if it can be derived from $A$ by deleting some characters, e.g. arms is a substring of algorithms. $C$ is a subsequence of $A$ if it can be derived from $A$ by deleting a prefix and postfix of $A$, e.g. gor is a subsequence of algorithms. Note that subsequences are a special case of substrings. The empty sequence $\emptyset$ and the entire sequence $A$ are always subsequences of $A$.

The Longest Common Substring problem is the following: given two strings $A[1..m]$ and $B[1..n]$ output the length of the longest $C$ that is simultaneously a substring of $A$ and $B$. The Longest Common Subsequence problem is defined analogously.

Assuming the two strings are represented in two one dimensional arrays $A[m]$, $B[n]$, the recursive function $\text{LCSubstr}(m, n)$ returns the length of the longest common substring of $A$ and $B$.

```c
int LCSubstr(int i, j)
    if i = 0 then return 0 endif;
    if j = 0 then return 0 endif;
    if $A[i] = B[j]$ then return $\text{LCSubstr}(i - 1, j - 1) + 1$;
    if $A[i] \neq B[j]$ then return $\max\{\text{LCSubstr}(i - 1, j), \text{LCSubstr}(i, j - 1)\}$;
```

(a) What can you say about the asymptotic running time of the above procedure.
(b) Give a more efficient algorithm for the longest common substring problem. Analyze the efficiency of your solution. (*) Optional for two extra points: Give an efficient algorithm that finds the longest common substring of three sequences.
(c) Give a more efficient algorithms for the longest common subsequence problem. Analyze the efficiency of your solution.

Problem 2. (20 = 10+10 points). Consider a set of $n$ intervals $[a_i, b_i]$ that cover the unit interval, that is, $[0, 1]$ is contained in the union of the intervals.

(a) Describe an algorithm that computes a minimum subset of the intervals that also covers $[0, 1]$.
(b) Analyze the running time of your algorithm.

(For question (b) you get credit for the correctness of your analysis but also for the running time of your algorithm. In other words, a fast algorithm earns you more points than a slow algorithm.)