**Solution of Part 1 of Assignment #2 (1 problem, 50 points)**
(Course: CS 401)

**Problem 1 (50 points):** A *string* $\mathcal{S}$ over an alphabet $\Sigma$ is a *concatenation* of some symbols from $\Sigma$. For example, if $\Sigma = \{a, b, c\}$ then both $abacaabca$ and $cbaaab$ are strings over $\Sigma$.

For two strings $\mathcal{S}$ and $\mathcal{T}$, we say that $\mathcal{T}$ is a *substring* of $\mathcal{S}$ if $\mathcal{T}$ can be obtained from $\mathcal{S}$ by *deleting* one or more symbols from $\mathcal{S}$. For example, if $\mathcal{T} = cac$ and $\mathcal{S} = babcbaabbccca$ then $\mathcal{T}$ is a *substring* of $\mathcal{S}$ since

$$\overbrace{\cancel{b}\ \cancel{a}\ \cancel{b}\ \boldsymbol{c}\ \cancel{b}\ \cancel{a}\ \boldsymbol{a}\ \cancel{b}\ \cancel{b}\ \cancel{c}\ \boldsymbol{c}\ \cancel{c}\ \cancel{a}}^{\mathcal{S}} \text{ is same as } \overbrace{cac}^{\mathcal{T}}$$

Given two strings $\mathcal{S} = s_1 s_2 \ldots s_n$ and $\mathcal{T} = t_1 t_2 \ldots t_m$ over some alphabet $\Sigma$, the goal of this problem is to design a greedy algorithm that decides in $O(m+n)$ time if $\mathcal{T}$ is a substring of $\mathcal{S}$. For this purpose, answer the following questions.

**(*a*) [15 points]** Describe a greedy algorithm that, given two strings $\mathcal{S} = s_1 s_2 \ldots s_n$ and $\mathcal{T} = t_1 t_2 \ldots t_m$ over some alphabet $\Sigma$, does the following:

- decides if $\mathcal{T}$ is a substring of $\mathcal{S}$ and outputs a "yes" or "no" response accordingly, and
- if $\mathcal{T}$ is a substring of $\mathcal{S}$ then shows which symbols of $\mathcal{S}$ are deleted to make it same as $\mathcal{T}$.

It suffices to describe the algorithm in pseudo-codes as long as sufficient details are provided. Justify why your algorithm runs in $O(m+n)$ time.

**(*b*) [35 points]** Prove that your greedy algorithm works correctly. For this, you must show both of the following:

**(b-1) [10 points]** if your algorithm outputs "yes" then $\mathcal{T}$ is a substring of $\mathcal{S}$, and

**(b-2) [25 points]** if $\mathcal{T}$ is a substring of $\mathcal{S}$ then your algorithm outputs "yes".

**Solution**: We give a greedy algorithm that finds the first character in $S$ that is the same as $t_1$, matches these two characters, then finds the first character *after this* in $S$ that is the same as $t_2$, and so on. The algorithm looks as follows (comments are enclosed with $(*$ and $*)$):

$(*\ k_1, k_2, \ldots$ denote the matches found so far $*)$
$(*\ i$ denotes the current position in $S$, $j$ denotes the current position in $T\ *)$
$i \leftarrow 1,\ j \leftarrow 1$
**while** $i \le n$ and $j \le m$ **do**
  **if** $s_i = t_j$ **then** $k_j \leftarrow i,\ i \leftarrow i+1,\ j \leftarrow j+1$ **else** $i \leftarrow i+1$
**endwhile**
**if** $j = m+1$ **then** return the subsequence $k_1, \ldots, k_m$
  **else** return "$T$ is not a substring of S"

The running time is $O(n)$: one iteration through the while look takes $O(1)$ time, and each iteration increments $i$, so there can be at most $n$ iterations.

It is also clear that the algorithm finds a *correct* substring if it finds any solution. It is harder to show that if the algorithm fails to find a substring, then no substring exists. Assume that $T$ is the same as the substring $s_{\ell_1}, \ldots, s_{\ell_m}$ of $S$. We prove by induction on $j$ that the algorithm will succeed in finding a substring and will have $k_j \leq \ell_j$ for all $j = 1, \ldots, m$. First consider $j = 1$. The algorithm lets $k_1$ be the first character in $S$ that is the same as $t_1$, so we must have that $k_1 \leq \ell_1$. Now consider the case when $j > 1$. Assume that $j - 1 < m$ and assume by the induction hypothesis that the algorithm has $k_{j-1} \leq \ell_{j-1}$. The algorithm lets $k_j$ be the first character in $S$ after $k_{j-1}$ that is the same as $t_j$ if such a match exists. We know that $\ell_j$ is such a match and $\ell_j > \ell_{j-1} \geq k_{j-1}$. Thus $s_{\ell_j} = t_j$, and $\ell_j > k_{j-1}$. The algorithm finds the first such index, thus we get that $k_j \leq \ell_j$.