Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Sandro Montanari
Tobias Pröger

17th April 2013

Datenstrukturen & Algorithmen Programming Exercise 8 FS 13

In this exercise we are going to implement a dynamic programming algorithm for solving the *longest common subsequence problem.* The problem is defined as follows. We are given two strings of text $A = a_1 \cdots a_n$ and $B = b_1 \cdots b_m$, where $a_1, ..., a_n, b_1, ..., b_m$ are characters from an alphabet Σ, and we look for a longest string that is a subsequence of both A and B. For example, if $A =$ "AGCAT" and $B =$ "GAC", the longest common subsequences would be one of the following: "AC", "GC", "GA".

The algorithm creates and fills a table $A[\cdot, \cdot]$ with $n+1$ rows and $m+1$ columns. For $0 \leq i \leq n$ and $0 \leq j \leq m$, an entry $A[i,j]$ represents the length of the longest common subsequence for the substrings of the original strings $a_1 \cdots a_i$ and $b_1 \cdots b_j$. Note that $i = 0$ and $j = 0$ represent the empty substrings. Thus, $A[i,0]$ and $A[0,j]$ are set to 0 for every i, $0 \leq i \leq n$ and for every j, $0 \leq j \leq m$. The other entries are computed as follows:

$$A[i,j] = \begin{cases} A[i-1,j-1]+1 & \text{if } a_i = b_j \\ \max\{A[i-1,j], A[i,j-1]\} & \text{otherwise.} \end{cases} \tag{1}$$

After the table has been filled, the entry $A[n,m]$ contains the length k of the longest common subsequence. The longest common subsequence is reconstructed from there using *backtracking.* If $a_n = b_m$, we set a_n as the k-th character of the longest common subsequence and we continue with the entry $A[n-1,m-1]$. If $a_n \neq b_m$, then a_n is not a character of the longest common subsequence. In this case we continue from $A[n-1,m]$ if $A[n,m] = A[n-1,m]$, and we continue from $A[n,m-1]$ if $A[n,m] \neq A[n-1,m]$. We stop once we have read all the k characters of the longest common subsequence.

Below, an example of the table $A[i,j]$ is shown for the strings $A =$ "ROCK" and $B =$ "ROLL".

$A[i,j]$	$\emptyset$	R	O	L	L
$\emptyset$	0	0	0	0	0
R	0	1	1	1	1
O	0	1	2	2	2
C	0	1	2	2	2
K	0	1	2	2	2

(2)

Input The first line contains only the number t of test instances. After that, we have exactly two lines per test instance. The first line contains the sequence A, and the second line contains the sequence B. Note that the alphabet used is $\Sigma = \{A, B, ..., Z\}$.

Output For every test instance we output only one line. This line contains the length of the longest common subsequence followed by the longest common subsequence computed by the above algorithm.

Example

Input:

```
2
AGCAT
GAC
ROCK
ROLL
```

Output:

```
2 AC
2 RO
```

Hint Use the code fragment

```
String A = scanner.next();
String B = scanner.next();
```

to read the complete strings A and B for each test instance. Furthermore, for a string `s`, you can use `s.length()` to determine its length, and `s.charAt(index)` to access the character of `s` located at the position `index`.

Hand-in: until Wednesday, 24th April 2013.