CSC 284/484 Advanced Algorithms - applied homework 1
due: April 5th midnight EST
(email source + output on the test input to the instructor)

This homework has different rules than the theoretical homework, most importantly, do not discuss any aspect of the applied problem with anybody else (except me), do not search for a solution online, do not use any written material when writing any part of the code (for example, no copy-paste, no open textbook when writing code, no reediting of an old source file from an old project, etc).

DEFINITIONS: Let $w \in \Sigma^*$ be word. The **length** of w is denoted by $|w|$, the i-th character of w is denoted $w[i]$ (we number the positions in w from 0 to $|w|-1$). A **subsequence** of w is a word $w[a_1]w[a_2]\dots w[a_k]$ where $0 \le a_1 < a_2 < \dots < a_k \le |w|-1$. The **reverse** of w is the word $w^R := w[|w|-1]w[|w|-2]\dots w[1]w[0]$. A word is called a **palindrome** if $w = w^R$. We let $w^1 := w$.

We are going to consider the following problem:

> Given a sequence of words $w_1, \dots, w_n \in \Sigma^*$ find $a_1, \dots, a_n \in \{1, R\}$ such that the word
> $$w_1^{a_1} w_2^{a_2} \dots w_n^{a_n}$$
> contains, as a subsequence, the longest possible palindrome.

For example if $n = 3$, $w_1 = abcd$, $w_2 = ocod$, and $w_3 = ab$, then $w_1^1 w_2 w_3^R = abcddocoba$ contains (as a subsequence) a palindrome of length 9 (abdocodba).

The problem can be solved using dynamic programming. Solve it and implement the solution. Your program should read the input from standard input (that is, if you execute it on a unix system you can feed it a file using `./a.out < file`). Your program should output the results to the standard output.

INPUT FORMAT:

- the first line contains an integer $n \in \{1, \dots, 1000\}$,
- the second line contains words $w_1, \dots, w_n \in \Sigma^*$ separated by spaces (we have $\Sigma = \{a, b, c, d, e, \dots, z\}$). Assume $|w_i| \le 5000$.

OUTPUT FORMAT:

- the first line contains the length of the longest palindrome,
- the second line contains the words $w_1^{a_1}, w_2^{a_2}, \dots, w_n^{a_n}$ separated by space (where $a_1, \dots, a_n \in \{1, R\}$ is the optimal solution),
- the third line contains the palindrome.

EXAMPLE INPUT:

```
3
abcd ocod ab
```

EXAMPLE OUTPUT:

```
8
abcd ocod ba
abdocodba
```

The objective of this section is to allow you to obtain sufficient problem-solving background to tackle the applied problem—solve but do NOT turn in. If you don't know how to solve any of the problems 1.1 to 1.8—ask your peers and/or ask for a solution in the problem session/office hours/email. Variation of one of the problems below can be included in Exam #3 or final. Each problem below can be solved using dynamic programming

1.1 We are given an $n \times n$ array A of zeros and ones. Give an algorithm to find the size of the largest contiguous all-ones square. Your algorithm must run in time $O(n^2)$.

1.2 We are given n positive numbers $a_1, \ldots, a_n$ (the numbers are not necessarily integers). The goal is to select a subset of the numbers with maximal sum and such that no three consecutive numbers are selected. Here are three example inputs together with optimal solutions (the numbers in boxes are selected):

$$\boxed{5}\ \boxed{5}\ 8\ \boxed{5}\ \boxed{5}$$

$$\boxed{5}\ 5\ \boxed{12}\ 5\ \boxed{5}$$

$$1\ \boxed{2}\ \boxed{2}\ 1\ \boxed{2}\ \boxed{1}\ 2\ \boxed{5}\ \boxed{5}$$

Give an $O(n)$-time algorithm for the problem.

1.3 We are given n positive integers $a_1, \ldots, a_n$ and another positive integer M. We want to figure out if we can select a subset of the integers which sums to M. Give an $O(Mn)$-time algorithm for the problem.

1.4 We are given n coin values $c_1, c_2, \ldots, c_n$ and an amount P (the c_i and P are positive integers). Unlike in the original coin change problem (where we had an unlimited supply of each coin value) we now have only 2 of each coin value. We would like to figure out whether we can pay P, and if we can, what is the minimal number of coins that can be used to pay P. Give an efficient algorithm for the problem.

For example if the coin values are $1, 2, 5, 6$ and $P = 15$ then the answer is yes - use 5 coins (since $15 = 6 + 6 + 2 + 1$ or $15 = 6 + 5 + 2 + 2$). (Note that we cannot pay $15 = 5 + 5 + 5$, since we have only 2 coins of value 5.)

1.5 Write a dynamic programming algorithm which for a given number n finds the smallest number of squares which sum to n (for example for $n = 7$ we need 4 squares ($7 = 2^2 + 1^2 + 1^2 + 1^2$), whereas for $n = 13$ we only need 2 squares ($13 = 3^2 + 2^2$)). Implement your algorithm and find all numbers from $\{1, 2, \ldots, 100\}$ which need 4 squares. Use "The On-Line Encyclopedia of Integer Sequences" to find a formula for the numbers which need 4 squares.

1.6 We are given a sequence of n positive numbers $a_1, \ldots, a_n$. Give an algorithm which finds the increasing subsequence[1] of $a_1, \ldots, a_n$ with the maximal sum. (For example on input $1, 101, 2, 3, 100, 4, 5$ your algorithm should output $1, 2, 3, 100$.)

1.7 Given a string $\overline{x} = x_1 x_2 \cdots x_n$ we would like to find the length of the longest palindromic subsequence[1]of $\overline{x}$ (a sequence is palindromic if it is the same as its reverse). Let $T[1..n, 1..n]$ be an array where $T[i, j]$ is the length of the longest palindromic subsequence of $x_i, \ldots, x_j$ (note that, $T[i, j]$ is undefined for $j < i$). Give an expression (or a piece of code) for $T[i, j]$ in terms of already computed values in T.

1.8 We have an $a \times b$ bar of chocolate (where a, b are integers). By breaking the bar we can either

- create two bars $a_1 \times b$ and $a_2 \times b$ where a_1, a_2 are integers and $a_1 + a_2 = a$, or
- create two bars $a \times b_1$ and $a \times b_2$ where b_1, b_2 are integers and $b_1 + b_2 = b$.

We can further break the resulting bars. Our goal is to 1) end up with bars that are square and 2) minimize the total number of breaks.

For example, if $a = 2$ and $b = 3$ then we use 2 breaks:

$$\begin{aligned} 2 \times 3 &\rightarrow \\ 2 \times 2 \text{ and } 2 \times 1 &\rightarrow \\ 2 \times 2 \text{ and } 1 \times 1 \text{ and } 1 \times 1&. \end{aligned}$$

For example, if $a = 2$ and $b = 4$ then we use 1 break:

$$\begin{aligned} 2 \times 4 &\rightarrow \\ 2 \times 2 \text{ and } 2 \times 2&. \end{aligned}$$

Give a dynamic programming algorithm which computes a table $T[1..a, 1..b]$ where $T[x, y]$ contains the minimal number of breaks to "squareize" an $x \times y$ bar.

[1] A subsequence of $a_1, a_2, \ldots, a_n$ is any $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$, where $1 \leq i_1 < i_2 < \cdots < i_k \leq n$. Thus, e. g., $1, 3, 5$ is a subsequence of $1, 2, 3, 4, 5$.