1. **Dance Dance Revolution** is a dance video game, first introduced in Japan by Konami in 1998. Players stand on a platform marked with four arrows, pointing forward, back, left, and right, arranged in a cross pattern. During play, the game plays a song and scrolls a sequence of \( n \) arrows (\( \leftarrow, \uparrow, \downarrow, \text{or} \rightarrow \)) from the bottom to the top of the screen. At the precise moment each arrow reaches the top of the screen, the player must step on the corresponding arrow on the dance platform. (The arrows are timed so that you’ll step with the beat of the song.)

You are playing a variant of this game called “Vogue Vogue Revolution”, where the goal is to play perfectly but move as little as possible. When an arrow reaches the top of the screen, if one of your feet is already on the correct arrow, you are awarded one style point for maintaining your current pose. If neither foot is on the right arrow, you must move one (and only one) of your feet from its current location to the correct arrow on the platform. If you ever step on the wrong arrow, or fail to step on the correct arrow, or move more than one foot at a time, or move either foot when you are already standing on the correct arrow, or insult Beyoncé, all your style points are immediately taken away and you lose.

How should you move your feet to maximize your total number of style points? For purposes of this problem, assume you always start with your left foot on \( \leftarrow \) and your right foot on \( \rightarrow \), and that you’ve memorized the entire sequence of arrows. For example, if the sequence is \( \uparrow \uparrow \downarrow \downarrow \leftarrow \rightarrow \leftarrow \rightarrow \), you can earn 5 style points by moving your feet as shown below:

![Diagram showing how to move feet to earn style points.]

Describe and analyze an efficient algorithm to find the maximum number of style points you can earn during a given VVR routine. Your input is an array \( \text{Arrow}[1..n] \) containing the sequence of arrows.

2. Recall that a palindrome is any string that is exactly the same as its reversal, like \( I \), or \( DEED \), or \( RACECAR \), or \( AMANAPLANACATACANALPANAMA \).

Any string can be decomposed into a sequence of palindrome substrings. For example, the string \( \text{BUBBASEESABANANA} \) (“Bubba sees a banana.”) can be broken into palindromes in the following ways (among many others):

\[
\begin{align*}
\text{BUB} \cdot \text{BASEESAB} \cdot \text{ANANA} \\
\text{B} \cdot \text{U} \cdot \text{BB} \cdot \text{A} \cdot \text{SEES} \cdot \text{ABA} \cdot \text{NAN} \cdot \text{A} \\
\text{B} \cdot \text{U} \cdot \text{BB} \cdot \text{A} \cdot \text{SEES} \cdot \text{A} \cdot \text{B} \cdot \text{ANANA} \\
\text{B} \cdot \text{U} \cdot \text{B} \cdot \text{A} \cdot \text{S} \cdot \text{E} \cdot \text{E} \cdot \text{S} \cdot \text{A} \cdot \text{B} \cdot \text{ANA} \cdot \text{N} \cdot \text{A}
\end{align*}
\]

Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string. For example, given the input string \( \text{BUBBASEESABANANA} \), your algorithm would return the integer 3.
3. Suppose you are given a DFA $M = (\{0, 1\}, Q, s, A, \delta)$ and a binary string $w \in \{0, 1\}^*$. 

(a) Describe and analyze an algorithm that computes the longest subsequence of $w$ that is accepted by $M$, or correctly reports that $M$ does not accept any subsequence of $w$.

*(b) [Extra credit] Describe and analyze an algorithm that computes the shortest supersequence of $w$ that is accepted by $M$, or correctly reports that $M$ does not accept any supersequence of $w$. (Recall that a string $x$ is a supersequence of $w$ if and only if $w$ is a subsequence of $x$.)

Analyze both of your algorithms in terms of the parameters $n = |w|$ and $k = |Q|$.
Describe and analyze an efficient algorithm to find the maximum number of style points you can earn during a given VVR routine. Your input is an array $\text{Arrow}[1..n]$ containing the sequence of arrows.
Describe and analyze an efficient algorithm to find the smallest number of palindromes that make up a given input string.
(a) Describe and analyze an algorithm that either computes the longest subsequence of $w$ that is accepted by $M$, or correctly reports that $M$ does not accept any subsequence of $w$.

*(b) [Extra credit] Describe and analyze an algorithm that either computes the shortest supersequence of $w$ that is accepted by $M$, or correctly reports that $M$ does not accept any supersequence of $w$. 