1. Let $X[0..n-1]$ be an array of characters. A subsequence of $X$ is a subset of not-necessarily-consecutive characters $\langle X[i_1], X[i_2], \ldots, X[i_k] \rangle$ such that $i_1 < i_2 < \cdots < i_k$. A subsequence can be empty (i.e., $k = 0$).

A palindrome is any string that is exactly the same as its reversal, like I, or DEED, or RACECAR, or AMANAPLANACATACANALPANAMA. Note that a palindrome may have an odd number of characters.

(a) Let $MaxPalSub(i, j)$ be the length of the longest subsequence of $X[i..j]$ that is also a palindrome. (For simplicity, we treat $X[i..j]$ as empty if $i > j$.) Fill in the blanks to complete the following recursive definition of $MaxPalSub(i, j)$.

$$MaxPalSub(i, j) = \begin{cases} 0 & \text{if } i > j \\ \_ & \text{if } i = j \\ 2 + MaxPalSub(i + 1, \_\_\_\_\_) & \text{if } i < j \text{ and } X[i] = X[j] \\ \max \{ MaxPalSub(\_\_\_, \_\_\_), MaxPalSub(\_\_\_\_, j) \} & \text{otherwise} \end{cases}$$

Advice: Suppose $j > i$. The longest palindrome in $X[i..j]$ should start with $X[i]$ and end with $X[j]$ if $X[i] = X[j]$. Otherwise, it must exclude at least one of $X[i]$ or $X[j]$.

(b) Use dynamic programming to write a method in Java that takes as its one parameter an array $x$ of characters and returns the length of the longest palindrome subsequence in $x$. Your method should be based on the above recurrence and run in $O(n^2)$ time given an array of length $n$.

(You can handwrite or type your code. Either way, it won’t actually be compiled.)

2. Suppose you are given a directed graph $G = (V, E)$ where each edge has an integer weight between 0 and some small value $M > 0$ along with a designated vertex $s \in V$. You may assume every vertex is reachable from $s$.

(a) What is the maximum possible distance a vertex can have from $s$?

(b) Describe how to implement Dijkstra’s algorithm so that it runs in $O(M|V| + |E|)$ time.

Advice: Instead of a priority queue, describe a different data structure.