Design Turing machines that solve the following problems. You do not have to write out the explicit program. Just describe the basic idea.

1. Design a Turing machine that inputs a word followed by a space and prints out a word (followed by a space) with all of the repeated letters removed. The output word should contain no more than one of each letter and they should appear in the same relative order as the first occurrence of these letters in the original word. If the input world is BOOKKEEPER, the output word should be BOKEPR.

Mark a space to the left of the word and go to the far right of the word, underline the last letter and go backward through the word to determine if that letter already exists. If it does not, go right to the underlined letter, remove the underline and try the previous letter. If the letter exists, go forward to the underlined and move every letter after the underlined one back one space, deleting the underlined letter. Then move to the previous letter. Process all letters in this fashion until you get to the far left of the string.

2. Design a Turing machine that accepts a word followed by a space, leaves the vowels (A,E,I,O,U) exactly where they are, but sorts the consonants relative to each other. For example, if the input were MICHIGAN, the output would be CIGHIMAN.

   This can be done by making several passes through the word. Each pass you move forward to the first consonant. From the first consonant you move forward to the next consonant, “remembering” the first consonant by virtue of a distinct set of states for each one. If the next consonant is greater than or equal to the first, make this consonant the first and look for the next beyond this one. You will either make it to the end of the word (meaning that all the consonants are in order) or you will find two adjacent consonants out of order. If you do find an out-of-order pair switch them, by writing the first in the second slot, and then going back to the first and writing the second one there. After switching a pair, go back to the beginning of the word for another pass. If nothing needs to be switched, go back to the beginning of the word and halt.

3. Design a Turing machine that accepts a word followed by a space and prints YES if the word contains at least two nonoverlapping palindromes of length at least 3, and NO if it does not. For example, if the input word were YABBADABBADOO, the output would be YES.

   Actually, all we really need to do is find palindromes of length 3 or 4. Go through the word until you find the first palindrome of length 3 or 4 which means you either look for an
ABBA pattern or an ABA pattern. (Come up with a series of states which checks whether a 3- or 4-letter palindrome exists starting at the current position and go through one position at a time until you find one.) Since the palindromes cannot overlap, start again at the location after the first palindrome and move forward one square at a time looking for another. If you find it, print YES. If you don't, print NO.