Task 1:

Write a function, `containsX(string)`, which searches through a string to see if the letter "X" exists. Do this without using the `in` operator, loops, the `count()`, or `contains()` methods.

The signature of this function is `(string) ---> boolean`.

Hint: Consider a string where it's really easy to figure out if X exists or not. Then, consider a way we can take the (arbitrarily long) input string and *do something* to it to get it closer to this base case.

Task 2:

Write a function, `count7(string)` that counts how many times the number "7" appears in a string. As before, solve this using recursion, and not loops, the count method, or similar.

The signature of this function is `(string) ---> int`.

Like before, start by thinking about the base case, and the recursive case, for this problem. For what string can we answer *immediately* without using any more advanced operators how many 7s it contains?

Task 3:

A *palindrome* is a string that reads the same forwards and backwards. Examples include the names "hannah" and "bob" and words such as "kayak" and "level". Write a function, `is_palindrome(string)` that determines whether a string constitutes a palindrome.

The signature of this function is `(string) ---> boolean`.

For this problem, consider when we can definitively say that a string is *not* a palindrome. Clearly, something like "hello" isn't, because it starts and ends with different letters. "leval" starts and ends with the same letter, but if we look one letter *in* from each end, we see a mismatch. Consider how you could translate this approach into base and recursive cases for this problem.