Read the code for the recursive function `count_odd`. And answer all 3 questions in this page and in the back.

1. def count_odd(lst):
   2.     
   3.         Return the number of odd numbers in <lst>
   4.     
   5.         :param lst: a nested list or int value
   6.         :type lst: int | list
   7.         :rtype: int
   8.         
   9.     if isinstance(lst, int):
   10.             return lst % 2
   11.     else:
   12.         for element in lst:
   13.             return count_odd(element)

Answer the questions below about the following client code.
>>> count_odd([1, [2, 6, 5], [9, [8, 7]]])

1. Based on the structure of the argument, state the three relevant recursive calls in this case, and what each one should return assuming `count_odd` is implemented correctly.

   - count_odd(1) should return 1
   - count_odd([2,6,5]) should return 1
   - count_odd([9, [8,7]]) should return 2

2. Now, assuming all recursive calls are correct, what recursive call(s) does this implementation of `count_odd` actually make, and what does the overall call to `count_odd([1, [2, 6, 5], [9, [8, 7]]])` actually return?

   The overall call should return 4, i.e. sum(1, 1, 2) or by list comprehension: sum([1, 1, 2])
3. Write a correct implementation of `count_odd`. (You may rewrite just the lines that should be changed.)

Replace Lines 11-13 with the following:

11. `else:
12.   return sum([count_odd(element) for element in lst])`