Louis's first procedure produces the desired list in reverse order because the `answer` is succesively put in the `cdr` position of a new pair each time `iter` is called. When the procedure reaches the last value in the original list, it squares it and places it in the `car` position of a pair, with the `cdr` pointing to the other squared values. The result is the original list squared but in reverse order.

Louis's fix means that nil becomes the `cars` of the first pair created, which is not the normal list structure of nested `cons` with a final `cdr` of nil signaling end-of-list. Additionally, each value in the list will be located in the `cdr` of each pair, instead of the `car`. The output of the test case in the scheme file confirms this. While the squared values do appear to be in the correct order, the paranthesis reveal that `cdr` of the result is 16, and to access the intended first value, you have to take the `cdr` of a successive number of `cars`.