LIST RECURSION

Consider the following procedure for summing up the items in a list:

\[
\begin{align*}
\text{(define (add-list lst)} & \\
& \quad \begin{cases} 
\text{if (null? lst)} & 0 \\
\text{(car lst)} & \text{(add-list (cdr lst))}
\end{cases}
\end{align*}
\]

**Question 1.** Using the Substitution Model, write out the result of evaluating the following:

\[
\begin{align*}
\text{(define foo (list 1 2 3 4))} \\
\text{(add-list foo)}
\end{align*}
\]

**Question 2.** We will now redefine add-list as follows. Perform the same exercise. Use the blank area next to the procedure definition to perform the Substitution Model.

\[
\begin{align*}
\text{(define (add-list lst)} & \\
& \quad \begin{cases} 
\text{(define (add-helper lst result)} & \text{result} \\
\text{if (null? lst)} & \text{(add-helper (cdr lst)} \quad (\text{result}}} \\
\text{(+ (car lst) result)))} & \text{())}
\end{cases}
\end{align*}
\]
Question 3. Write a procedure to count the number of items in a list. Call it length.

Question 4. What does the following procedure do? 
E.g. if you evaluate (mystery '(foo bar) '(blatz cat dog)), what will it produce? Hint: Use the Substitution Model.

(define (mystery lst1 lst2)
  (if (null? lst1)
      lst2
      (cons (car lst1)
            (mystery (cdr lst1) lst2)))))