Problems on sequences

1. Give formal recursive definitions for the following operations on sequences. It is assumed that the *push* operation adds an element to the beginning of the sequence. The ith element in the sequence is the element that has exactly i elements occurring before it. Thus $head(S)$ is the 0th element in the sequence.

   (a) $insert(S, x, i)$: inserts the element x in the sequence S after exactly i elements, if $0 \leq i \leq length(S)$, otherwise undefined.

   (b) $find(S, i)$: returns the ith element in the sequence if $0 \leq i < length(S)$ and is undefined otherwise.

   (c) $erase(S, i)$: removes the ith element in the sequence if $0 \leq i < length(S)$, otherwise does nothing.

   (d) $swap(S, i, j)$: swaps the ith and jth element in the sequence if $0 \leq i < j < length(S)$, otherwise does nothing.

2. Consider an abstract data type T defined as follows. There is a value called λ in T. If t_1 and t_2 are two values in T then $t_1 \cdot t_2$ is also a value in T, where $\cdot$ is an operation defined on values of type T. If a set of values S contains λ, and if for all $t_1, t_2 \in S \cap T$, $t_1 \cdot t_2 \in S$ then $T \subseteq S$. Further $t_1 \cdot t_2 \neq \lambda$ for all $t_1, t_2 \in T$ and $t_1 \cdot t_2 = t_1' \cdot t_2'$ iff $t_1 = t_1'$ and $t_2 = t_2'$. Give at least 3 different examples of types that satisfy these axioms. What does the $\cdot$ operation mean in each case? This is similar to numbers except that instead of *next*, we have a binary operation $\cdot$. There are at least 200 different kinds of objects that satisfy these axioms.

   Consider the function f defined on this type.

$$
\begin{aligned}
f(\lambda) &= 0 \\
f(t_1 \cdot t_2) &= 1 + \max(f(t_1), f(t_2)))
\end{aligned}
$$

   In your example types, what does this function mean?

3. The set of bit strings is defined as follows. λ is a bit string, and if S is a bit string then $S \cdot 0$ and $S \cdot 1$ are also bit strings. The induction axiom also holds. Consider the following functions defined on bit strings.

$$
\begin{aligned}
even(\lambda) &= true \\
even(S \cdot 0) &= even(S) \\
even(S \cdot 1) &= !even(S) \\
f(\lambda) &= \lambda \\
f(\lambda \cdot 0) &= \lambda \cdot 1 \\
f(\lambda \cdot 1) &= \lambda \cdot 0
\end{aligned}
$$

$$
\begin{array}{l}
\text{For all } S \neq \lambda \\
\qquad \begin{aligned}
f(S \cdot 0) &= S \cdot 1 \text{ if } even(S) \\
&= f(S) \cdot 0 \text{ otherwise.} \\
f(S \cdot 1) &= f(S) \cdot 1 \text{ if } even(S) \\
&= S \cdot 0 \text{ otherwise.}
\end{aligned}
\end{array}
$$

Prove the following properties of the function f.

(a) f is one-to-one, that is $f(S_1) = f(S_2)$ if and only if $S_1 = S_2$.

(b) f is onto, that is for every string S_1, there is a string S_2 such that $f(S_2) = S_1$.

(c) Give a recursive definition of the inverse function of f, that is a function g such that $g(f(S)) = S$ for all bit strings S.

(d) Let $f^0(S) = S$ and $f^{k+1}(S) = f(f^k(S))$. Prove that for any string S and any symbol 0 or 1, $f^{2k}(S \cdot x) = f^k(S) \cdot f^k(x)$, where $f(0) = 1$ and $f(1) = 0$.

(e) For any string S, $f^k(S) = S$ if and only if k is a multiple of $2^{length(S)}$.

This function f defines what is called the 'Gray code' for bit strings.

4. A sequence S is said to be obtained by interleaving sequences S_1 and S_2 if S can be partitioned into two subsequences that are equal to S_1 and S_2. In other words, each element of S must be placed in exactly one of the two subsequences, keeping the order the same as in S, and the resulting subsequences are S_1 and S_2. Define a function $interleave(S_1, S_2, S)$ that returns true iff S can be obtained by interleaving S_1 and S_2. Given a sequence S, can you find the number of pairs of sequences S_1, S_2 such that S can be obtained by interleaving S_1 and S_2? Given S_1, S_2, can you find the number of different sequences S, that can be obtained by interleaving S_1, S_2?