Write a program to sort a list. Execution time should be at most $n \times \log n$ where $n$ is the length of the list.

There are many ways to sort in time $n \times \log n$. I'll do merge sort. Let the list variable be $L$. Let $i$, $j$, and $k$ be natural variables. Define specifications $S$ (for Sort) and $T$ as follows.

\[
S = (\forall a, b: 0, \ldots, \#L \cdot a \leq b \Rightarrow L'[a] \leq L'[b]) \land \text{perm } L' \ L
\]
\[
T = (\forall a, b: i, \ldots, k \cdot a \leq b \Rightarrow L'[a] \leq L'[b]) \land \text{perm } (L'[i], \ldots, k) \land (L[i], \ldots, k) \land L'[0, \ldots, i] = L[0, \ldots, i] \land L'[k, \ldots, \#L] = L[k, \ldots, \#L]
\]

\[
\text{perm } A \ B = \forall x: \varphi(\forall i: 0, \ldots, \#A \cdot A \ i = x) = \varphi(\forall i: 0, \ldots, \#B \cdot B \ i = x)
\]

I have just realized that top-down mergesort (mergesort both halves of the list, then merge the two sorted halves) will require a stack of values, either as parameters (Chapter 5) or as an explicit stack (Chapter 7). So I'll try bottom-up mergesort (merge pairs of singles, then pairs of pairs, then pairs of 4s, and so on). UNFINISHED