# Quiz 10: Functional Specifications

● Graded

**Student**

Abhinav Shripad

**Total Points**

**3.5 / 10 pts**

**Question 1**

## Abstract Data Type of Sorted Sequences

**3.5** / 10 pts

- ✓ **+ 1 pt** present(e,emptyseq()) = false
- **+ 1 pt** present(e,singleton(e)) = true
- **+ 1 pt** present(f,singleton(e)) = false if (f =/= e) else true
- **+ 1 pt** present(e,merge(x,y)) = present(e,x)| present(e,y)
- **+ 1 pt** first(emptyseq()) = None
- **+ 1 pt** first(singleton(e)) = e
- **+ 1 pt** first(merge(x,y)) = min(first(x),first(y))
- ✓ **+ 1 pt** merge(x,emptyseq()) = merge(emptyseq(),x) = x
- ✓ **+ 2 pts** merge(merge(singleton(e),x),merge(singleton(f),y)) = merge(merge(singleton(e), singleton(f)), merge(x,y))
- **+ 0 pts** Incorrect/not answered
- 💬 **− 0.5 pts** Point adjustment

## Q1 Abstract Data Type of Sorted Sequences

**10 Points**

Assume that we have a data type *elem* which is totally ordered, i.e., there is an operation $\leq : elem * elem \rightarrow \mathtt{bool}$.

Our task is to specify a data type $elem\ \mathtt{ordseq}$ of *ordered sequences*, i.e., sequences $l$ such that if $x$ appears before $y$ in $l$ then $x \leq y$.

The operations that are to be supported are:

- $emptyseq : \mathtt{unit} \rightarrow elem\ \mathtt{ordseq}$, creates an empty sequence
- $present : elem \rightarrow elem\ \mathtt{ordseq} \rightarrow \mathtt{bool}$, returns true if the element is in the ordered sequence
- $singleton : elem \rightarrow elem\ \mathtt{ordseq}$, creates a singleton ordered sequence containing the given element.
- $first : elem\ \mathtt{ordseq} \rightarrow elem$, returns the first element of the ordered sequence
- $merge : elem\ \mathtt{ordseq} \rightarrow elem\ \mathtt{ordseq} \rightarrow elem\ \mathtt{ordseq}$, merges two ordered sequences to create an ordered sequence.

Write (conditional) equational specifications for the abstract data type of sorted sequences.

```
emptyseq() = [].
singleton(x) = [x].
present(x, []) = false
present(x, [y:ys]) = (x = y) or present(x, ys).
first([x:xs]) = x.
merge(x,[])= x.
merge([x:xs],[y:ys]) = if(x=<y) then [x:merge(xs,[y:ys])]. else [y:merge([x:xs],ys)].
```