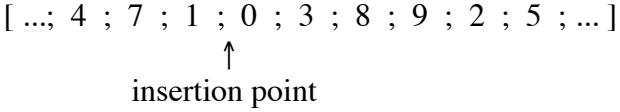
448 (insertion list) An insertion list is a data structure similar to a list, but with an associated insertion point.



insert puts an item at the insertion point (between two existing items), leaving the insertion point at its right. *erase* removes the item to the left of the insertion point, closing up the list. *item* gives the item to the left of the insertion point. *forward* moves the insertion point one item to the right. *back* moves the insertion point one item to the left.

(a) Design axioms for a doubly-infinite data-insertion list.
(b) Design axioms for a doubly-infinite program-insertion list.
(c) Design axioms for a finite data-insertion list.
(d) Design axioms for a finite program-insertion list.

After trying the question, scroll down to the solution.

(a) Design axioms for a doubly-infinite data-insertion list.

(b) Design axioms for a doubly-infinite program-insertion list.

§ Here is a weak theory.

$item'=x \Leftarrow insert\ x$

$item'=item \Leftarrow F \vee (back.B.forward)$

$forward.back = back.forward = ok$

$F \Leftarrow ok \vee ((\exists x\cdot insert\ x) \vee forward.F.erase \vee back) \vee (F.F)$

$B \Leftarrow ok \vee (\exists x\cdot insert\ x) \vee erase \vee (back.B.forward) \vee (B.B)$

Here is a strong theory.

$ok = F \wedge B = forward.back = back.forward = insert\ x.erase$

$insert\ x = (back.F) \wedge item'=x \wedge B$

$F = ok \vee ((\exists x\cdot insert\ x) \vee forward.F.erase \vee back) \vee (F.F)$

$B = ok \vee (\exists x\cdot insert\ x) \vee erase \vee (back.B.forward) \vee (B.B)$

(c) Design axioms for a finite data-insertion list.

(d) Design axioms for a finite program-insertion list.