should return

$$@_1\langle 3, @_1\langle 2, @_1\langle 1, @_1\langle 0, @_0\langle\rangle\rangle\rangle\rangle\rangle$$

which is often written as

$$3 :: 2 :: 1 :: 0 :: nil$$

representing the list of 3, 2, 1, 0. Note that these are all the possible outcomes of the parameter function to backtrack. To implement backtrack with callcc and throw, we need a few more features in our language.

⟨exp⟩ ::= mkref ⟨exp⟩ — allocates a memory cell, initialises it with ⟨exp⟩ and returns the reference to the cell.

| val ⟨exp⟩ — dereferences the reference

| ⟨exp⟩ := ⟨exp⟩ — updates a reference

| ⟨exp⟩ =ref ⟨exp⟩ — reference equality check.

Syntactic sugar.

$nil \stackrel{def}{=} @_0\langle\rangle$

$e :: e' \stackrel{def}{=} @_1\langle e, e'\rangle$

listcase e of $(e_1, e_2) \stackrel{def}{=}$ sumcase e of $(\lambda v.\, e_1,\ \lambda v.\, ((e_2\ v.0)\ v.1))$

let $v \equiv e$ in $e' \stackrel{def}{=} ((\lambda v.\, e')\ e)$

$e ; e' \stackrel{def}{=}$ let $v \equiv e$ in e' (for fresh v)

backtrack $\stackrel{def}{=} \lambda f.$ let $rl \equiv$ mkref nil in
let $cl \equiv$ mkref nil in
$rl := f\,(\lambda u.\, \text{callcc}\,(\lambda k.\, (cl := k :: \text{val } cl);$
true))
$:: \text{val } rl\ ;$
listcase (val cl) of
(val rl,
$\lambda c.\, \lambda r.\, (cl := r\ ;$ throw c false))