



A solution to the exercise

```
// Binary-search
// If key is a member of list lst then return True, otherwise False
// For demonstration purposes each number is no greater than 8 bits long

// ff finds the maximum index which is no greater than 255
ff : {a, b, c} (fin c, c >= width a, c >= 8) => [a]b -> [c]
ff lst = if (length lst) > 255 then 255 else (length lst)-1

bsearch : {a,b} (Cmp a, 8 >= width b) => (a, [b]a) -> Bit
bsearch (key, lst) = search (0, (ff lst), w)
  where
    w = length lst
    // The function "search" is not symbolically terminating,
    // so we use an "extra" recursion counter (c) for termination purposes
    // Note that this is only needed when doing the proof.
    search : {d} (fin d, d >= 1) => ([8], [8], [d]) -> Bit
    search (low, high, c) =
      if (c == 0) \ / (low > high) \ / (high > w) then False
      else if midVal < key then search (mid+1, high, c-1)
      else if midVal > key then search (low, mid-1, c-1)
      else True
      where
        midVal = lst @ mid
        mid = computeMid (low, high)

// Good version
computeMid : ([8], [8]) -> [8]
computeMid (low, high) = (low + (high-low) / 2)

lst3 = [0, 2, 3, 5, 6, 8, 9, 10, 11, 12, 13, 14, 16, 18,
        20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 33, 34,
        35, 36, 37, 39, 40, 41, 32, 43, 44, 45, 46, 47, 49, 50,
        51, 52, 53, 54, 56, 58, 59, 60, 61, 62, 63, 65, 66, 67,
        68, 69, 70, 71, 72, 73, 74, 76, 77, 78, 79, 80, 81, 82,
        84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97,
        98, 99, 101, 102, 103, 104, 105, 106, 107, 108, 110, 112, 113, 114,
        116, 117, 119, 120, 121, 122, 124, 125, 126, 127, 129, 130, 131, 132,
        133, 134, 135, 136, 138, 139, 141, 143, 144, 145, 147, 148, 149, 150,
        152, 153, 155, 156, 157, 158, 159, 160, 162, 163, 164, 166, 167, 168,
        169, 170, 172, 174, 175, 176, 177, 178, 179, 181, 182, 183, 184, 185,
        187, 188, 189, 190, 191, 192, 197, 198, 199, 200, 201, 202, 203, 204,
        206, 207, 208, 209, 210, 211, 213, 214, 215, 216, 218, 220, 221, 222,
        223, 224, 226, 228, 229, 230, 231, 232, 234, 235, 236, 237, 238, 239,
        240, 241, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 255]
: [210][9]
```

```

// Reference spec for search ("obviously" correct..)
refSearch : {n,a} (fin n, fin a) => ([a], [n][a]) -> Bit
refSearch (key, lst) = [ key == x | x <- lst ] != zero

// bsearch and refSearch are equivalent when the input is
// onlyIncreasing, so recognize such sequences:
onlyIncreasing : {a,b} (fin a, fin b) => [a][b] -> Bit
onlyIncreasing lst = pairComps == ~zero
  where pairComps = [ x < x' | x <- [0] # lst | x' <- lst ]

// The theorem holds, establishing correctness of bsearch (key, lst).
bsearchOK : [8] -> [100][8] -> Bit
property bsearchOK key lst =
  if onlyIncreasing (lst)
  then bsearch (key, lst) == refSearch (key, lst)
  else True

// Make tests of bsearch and refSearch
try key lst = bsearch (key, lst)
ref key lst = refSearch (key, lst)

// Cryptol> :l csearch_nobug.cry
// Loading module Cryptol
// Loading module Main
// Main> :prove bsearchOK
// Q.E.D.
// (Total Elapsed Time: 6m:15.330s, using "Z3")
// Main>

```