EQUALITY AND LIST MEMBERSHIP

Here are simplified explanations of eq?, eqv? and equal?:

- eq? is true if the two things being compared are the same object, or if they are the same symbol. This is the most discriminating test. Note: (eq? 3 3) is unspecified.
- eqv? uses eq? for symbols and booleans, uses = for numbers, uses string=? for strings, and uses eq? for pairs.
- equal? recursively tests objects in parallel structures. It uses eqv? to test non-pairs (single items), and then if there is a list structure, it recurses on the car and cdr, using equal?.

Assume the following Scheme expressions are evaluated, in order, and then tested for the various equivalences.

**Question 1.** Fill in the table below with “true,” “false,” or “unspecified” based on the use of eq?, eqv? and equal? to compare the items in the first two columns.

```
(define a (list 1 2))
(define b '(1 2))
(define c a)
(define d (car a))
(define e 1)
```

<table>
<thead>
<tr>
<th></th>
<th>eq?</th>
<th>eqv?</th>
<th>equal?</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Scheme provides the following primitives to test whether an item is a member of a list:¹

\[
\text{(member } v \text{ lst)} \quad \text{Locates the first element of } \text{lst} \text{ that is } \text{equal?} \text{ to } v. \text{ If such an element exists, the tail of } \text{lst} \text{ starting with that element is returned. Otherwise, the result is } \#f. \\
\text{(memv } v \text{ lst)} \quad \text{Like } \text{member}, \text{ but using } \text{eqv?}. \\
\text{(memq)} \quad \text{Like } \text{member}, \text{ but using } \text{eq?}
\]

**Question 2.** Write a definition for `member`. It should be an iterative recursion that has O(N) performance in time, and O(1) performance in space.

¹ https://docs.racket-lang.org/reference/pairs.html