CMSC 15100: HW6 Problem 1

Victor Cheung 438907

0.1 Time Cost of `set-mem?`

```
;; a symbol-set is a (listof symbol) containing no duplicate symbols

;; set-mem? : symbol symbol-set -> bool
(define (set-mem? s ss)
  (cond
    [(empty? ss) false]
    [else (or (symbol=? s (first ss))
              (set-mem? s (rest ss)))]))
```

Everything in `set-mem?` is an inexpensive constant time operation except for recursive calls to `set-mem?`.
We choose as our cost model applications of `symbol=?`.
We define a cost estimation model:

$$C_s : nat -> nat$$

Where the input is the size of the input `symbol-set` and the output is the time cost per our model (applications of `symbol=?`).
By inspection, we know that

$$\begin{aligned}
& C_s(0) = 0 \\
\text{for } n > 0,\quad & C_s(n) = 1 + C_s(n-1) \\
& C_s(n) = 1 + 1 + C_s(n-2) \\
& C_s(n) = 1 + 1 + \ldots + C_s(0) \\
& C_s(n) = \sum_{i=1}^{n} 1 = n
\end{aligned}$$

Then `set-mem?` has linear time cost $O(n)$.

0.2 Time Cost of `list->sym-set`

```
;; insert : symbol symbol-set -> symbol-set
;; insert is the same time cost as set-mem?
(define (insert s ss)
  (if (set-mem? s ss) ss (cons s ss)))

;; list->sym-set : (listof symbol) -> symbol-set
(define (list->sym-set ss)
  (cond
    [(empty? ss) empty]
    [else (insert (first ss)
                  (list->sym-set (rest ss)))]))
```

Everything in `list->sym-set` is an inexpensive constant time operation excepting recursive calls to `list->sym-set` and calls to `insert`.
We choose as our time cost model applications of `insert`.
Our cost model function is

$$C_l : nat- > nat$$

Where the input is the size of the `(listof symbol)` argument and the output is the time cost per our model (applications of `insert`).
By analysis we see that

$$\begin{aligned} C_l(0) &= 0 \\ \text{for } n > 0,\ \ C_l(n) &= C_s(n-1) + C_l(n-1) \end{aligned}$$

Note that `insert` has the same time cost as `set-mem?`. Then C_s represents its time cost. In which case we substitute such that

$$\begin{aligned} C_l(n) &= (n-1) + C_l(n-1) \\ C_l(n) &= (n-1) + C_s(n-2) + C_l(n-2) \\ C_l(n) &= (n-1) + (n-2) + C_l(n-2) \\ C_l(n) &= (n-1) + \ldots + 0 + C_l(0) \\ C_l(n) &= \sum_{i=1}^{n-1} i = \frac{n \cdot (n-1)}{2} \\ C_l(n) &= \frac{n^2}{2} + \frac{n}{2} \end{aligned}$$

Since n^2 is the dominant term then `list->sym-set` has quadratic time cost with $O(n^2)$.