Scott Waldron W00970369
Oct 11,2016

1. Give a brief description of the function insertR.

insertR takes three arguements: 2 atoms and 1 list (new old and lat). insertR appends new to old and returns the new list.

2. Give a brief description of the function insertL.

insertL takes three arguements as well: 2 atoms and 1 list (new old and lat). insertL prepends new to old and returns the new list.

3. What are some differences between the functions insertR and insertL?

insertR and insertL will both recursively look through a list one atom at a time and ask whether or not the atom equals old. The difference is after that boolean where insertL will prepend the word and insertR with append the word. There is no replacing, just adding to the left or to the right.

4. Give a brief description of the function subst.

subst takes three arguements: 2 atoms and 1 list (new old and lat). subst will find the first occurance of old and replace it with new.

5. Give a brief description of the function subst2.

subst2 takes four arguements: 3 atoms and a list (new o1 o2 and lat). subst2 will find the first occurance of either o1 or o2 and replace it with new.

6. What are some differences between the functions subst and subst2?

They both work very similarly except subst2 will find the first occurance of either o1 or o2 and replace it with new while subst is only given one atom to find and replace with new.

7. What does the function member? return?

member? will return a #t or #f based on whether the arguement (atom) was found in the list.

8. What does the function rember return?

rember will return a list with the first occurance of the the atom removed.

9. Give a brief description of the function firsts.

firsts is a function that will take one arguement (l) and construct a list of the first S-expressions in each internal list.

10. Provide your own function definition below. What does your function accept as arguments and what does it return? Please give a sample Scheme use of your function.

removeLast is a function that will remove the last S-expression from a list and
return the list without the final element. It takes only one arguement: the list.

EXAMPLE:

```
        (define removeLast
                (lambda (l)
                        (cond
                                ((null? (car l)) (quote()))
                                (else (cons (car l)
                                        (removeLast (cdr l)))))))
```

Thank you! Have a great week!