Kelvin Mei
COSI121B
PS2 Writeup

2.4

```
(define (cons x y)
  (lambda (m) (m x y)))

(define (car z)
  (z (lambda (p q) p)))
```

cdr is basically like car, except instead of returning p, it returns q.

Substitution method:

```
(car (cons x y)
(car (lambda m (m x y)))
((lambda (p q) p) x y)
= x
```

Thus, to get y as the solution ("the cdr"), we can just change (p q) p to (p q) q.

2.22

This answer list is in the reverse order of the one desired because of the way the user processes the iteration. It processes the first element and the merge it with the empty list. Then it processes the second element and merge it with the new list, but it puts the second element before the first one. It then repeats until the method is over, resulting in the reverse of the desired output.

By swapping the argument in the cons, the order comes out right. However, that produces a very undesirable outcome. The arguments passed to the cons is a list and a value. That will result in multiple parentheses and periods that occur. In order to have a single parentheses list, cons requires the arguments as a (cons <value> <list>), not the other way around. (square-list (list 1 2 3 4)) results in (((((() . 1) . 4) . 9) . 16).

2.25

```
(car (cdaddr (list 1 3 (list 5 7) 9)))

(caar (list (list 7)))

(cadadr (cadadr (cadadr (list 1 (list 2 (list 3 (list 4 (list 5 (list 6 7))))))))))
```

2.26

(append x y) should result in (1 2 3 4 5 6), since it merges the lists together into one.

(cons x y) should result in ((1 2 3) 4 5 6), since it places argument one inside argument two, make a list inside a list.

(list x y) should result in ((1 2 3)(4 5 6)), since you're creating a list which takes two lists as arguments.

2.27

The nil method defines the empty list.

The atom? method checks if the input is a list or not.

In deep-reverse, I do the same thing as reverse except I check if the (car things) is a list or not. If it's a list, I recursively call the deep-reverse on the list. If not, I just cons it to the answer.

2.28

The nil and atom? method remains from the previous problem.

This one is merely an edit on 2.27, where the output order is swapped and the cons are changed to appends to create one single list.

Lab

Q1: Simply adding English phrases to the list. Nothing special.
Q2: Adding the pairs to the replacement-pairs will not work because only some of the pairs will be changed. The error in this is that the replaces are looping. For example, if there is a 'you' and it gets replaced by an 'i', scheme will then read the 'i' and change it back to 'you', causing some of the replacements to not work. To combat this problem, I had two sets of replacement pairs. The first set changes the words to temporary placeholders like t1, t2, t3. After the words in the phrase are all changed, I took the new phrase and I converted the new words back to real words. This prevents the looping problem before.
Q3: I added a record list to the doctor-driver-loop. When visit-doctor is called, it sends a nil list to the loop and starts a record of what has been said. Every sentence said will be appended to that record using (append (list record)(list user-response)). (list record) and (list user-response) is necessary to make sure the list is kept by sentences, not by words.
Lastly, in the reply function, I pass in the record list. It calls (prob 5 10), which is 50% chance after the initial (fifty-fifty). If that conditional is true, it appends '(earlier you said that) to one of the phrases said before. This phrase is chosen randomly, using the length of the record – 1 to account for the nil list passed in at the beginning. The functions used here are (length list) to find the length of a list and (list-ref list index) to get the list item at the index.
Q4: In this problem, I modified visit-doctor to accept a number. If the number is 0, it will simply reply closing time and end the loop. If the name is suppertime, it will reply time to go home. It then passes the number into the doctor-drive-loop where whenever a patient says goodbye, the number is deducted by 1. Then the number is passed back into the visit-doctor loop with a new name which is received using (ask-patient-name). Once number reaches zero, it will reply closing time and end the loop.