Samuel Bernheim (bernheim@brandeis.edu) 2016-2-25

Problem 2.22

The reason the list is in reverse order is because when cons is used to bind (square (car things) and answer, answer, after the first iteration of the list is (1 . nil). Then when cons is called to bind 4 and answer answer becomes 4 1 nil. Then on the next pass through (car things) will return 3, (square 3) will return 9 and cons will then bind 9 to answer which was 4 1 and so answer will now be 9 4 1. This process will continue until things is an empty list. Essentially the most recent number is aded to the front of the list because the order of the cons is first (square (car things)) and then answer.

Problem 2.25
(cdr (car (cdr (cdr z))))

Problem 2.26

(append x y) will return the list (1 2 3 4 5 6)

(cons x y) will return ((1 2 3) 4 5 6)

(list x y) will return ((1 2 3) (4 5 6))

Problem 2.27

For this problem I started by writing a procedure which will return the last element in a list. I then wrote a procedure which gets the size of a list, and a procedure which will reverse the elements of a list, for example ((1 2 3) (4 5 6)) would become ((4 5 6) (1 2 3). This reverse procedure works using the previous two procedures to test when the list has been reversed and finding the last element in a list and attaching it to the new list. Then I wrote the deep reverse procedure using the reverse procedure.

LAB

Question 1

I added the phrases (how is this negative), (what makes you feel), (where does this come from) and (I can't help you).

Question 2

Originally, (change-person '(you are not being very helpful to me)) returned: (you are not being very helpful to you), but adding the additional pairs to the change-person procedure did not result in any change. After I changed the order of the pairs in the change-person procedure the output changed to: (i am not being very helpful to you). The bug which occurs is that once a change is made from you to i for example when run again the phrase will be changed back from i to you since currently the replacement procedure is running like a double for loop and changing words multiple times instead of just once. To remedy this I rewrote the replace and the many-replace procedures. The new replace procedure works by first checking if the replacement-pairs list is null. If so the target which would have been replaced is just returned. Otherwise if replacement-pairs is not null, I get the (car (car replacement-pairs)) which is the word I am looking for which will be replaced and seeing if it is equal to the target which could be replaced. If these two words are equal, then I get the corresponding word from the replacement-pair pairing and return that value. Otherwise I call replace passing in the same target and the cdr of replacement-pairs to see if the target word matches any other patterns in the replacement-pairs list. Many-replace now takes a list of replacement-pairs and a sentence and works in the following way. If the replacement-pairs list is null then the sentence is returned. If sentence is null sentence is returned. Otherwise a cons is made between (replace (car sentence) replacement-pairs). This simply calls replace on the first word of the sentence and the replacement-pairs list. Many-replace is then called on the cdr of sentence (so now word is changed more than once) and the same replacement-pairs list.

Question 3

To store all the responses I made the doctor-driver-loop take in an additional parameter called lst which will be a list of all responses. On each recursive call of the loop I pass in for this lst parameter the cons of lst and user-response adding the user-response to lst. Then to use this list of user responses I wrote several procedures. One for choosing a random number between 0 and the size of the list, one for then choosing the response given an index. Then to the reply procedure I made it randomly choose between one of three options, two of which are the hedge or qualifier procedures. The new procedure now will test to ensure the lst of user responses is not null and will then call append to the list earlier you said that and a random phrase from the lst of user-responses.

Question 4

To handle the case of a user entering the keyword suppertime, I added an extra condition to the driver-loop saying if the user-response equaled suppertime, then perform the corresponding actions. For the automatic continuation for the doctor I made it so that upon entering goodbye, then the appropriate phrases will be printed and then auto-start (a procedure I wrote) gets executed. Auto-start takes in as a parameter the number of patients the doctor can see. If that number is 0, then the phrase done for the day is printed to the console. If not, then another procedure called continue (which I also wrote) is called. Continue which takes the number of patients the doctor can see as a parameter, will then print who are you, read in the users name, and then call visit doctor passing the appropriate parameters.