Notes, Questions and answers from previous terms’ Piazza:

# The iterator problem:

# Scheme: Returning a value AND mutating the slist.

>(define lst `(a b c d))

How do I return the car of a, but also set! lst  to the cdr of lst all in one procedure call?

**the students' answer,**

*where students collectively construct a single answer*

Let blocks can have multiple expressions in them, and the result of the last expression will be the result of the whole block. For example:

(let ((a (car lst)))

(set! lst (cdr lst))

a)

**the instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/jl863803n0a6tl?cid=53)

Cons, lambda, letrec, begin, and the individual cases of cons and case can also have multiple expressions. They get executed in order, and the value of the last expression is returned.

# Subst-leftmost problem:

# subst-leftmost: the main indicators that you are violating the rules (instructor note)

**Summary of the rules:**

1. Don't go through any sublist twice

2.  If you did a substitution in the car, do not call the recursive procedure on the cdr,

Some things that will make me suspect that you may have violated the rules:

1. You call **equal?**

2.  You call **list?**

3. You call **append**, where the first argument can have length that is greater than 2.

4, You call your recursive procedure on the cdr without first checking to see if you made a substitution in the car.

If you do one of these, I will look closely at your code; most likely you will lose about 10 points for list? , 25 points for equal? and 15 for append

Don't forget to **FOLLOW THE GRAMMAR**.  For s-lists, that means that there are three main cases:  list is empty, car is a symbol, car is a list (this can be the else case).  See the slist examples i the live-in-class folder.

# A8: - make-slist-leaf-iterator MUST use my stack class

I want you get experience in both creating a new class and using an existing class.  So the stack you use must be created by calling (make-stack).  You are allowed to add new methods to that class, but I doubt that you will have to so so.

# existence of an answer for subst-leftmost

Can we assume that there always will be at least one symbol in the list that we need to substitute? For example, if we need to substitute a with b, will there always be b in the list?

**Students’ answer:**

No.  Take a look at the first test case:

(subst-leftmost 'k 'b '() eq?)

The expected answer is the empty list.

# Ideas for using stacks to reach leaves of sublist in A8 Problem 1?

I'm trying to figure out how to use stacks to iterate through the elements of a sublist.

I have written a sublist out as a binary tree where the left child is the car and the right child is the cdr and can see why the leaves of such a tree would return the elements of a sublist in order, but I can't figure out how I'm supposed to use stacks to get to these leaves.

I tried an approach similar to a preorder traversal, but I had trouble figuring out how much I should push / pull to know which combination of cars and cdrs I am in the sublist.

Am I using the right approach for this problem? What should I be doing differently?

**Students’ answer:**

You can start by pushing the entire s-list onto the stack. Each iteration you pop the top item off. If the car is a symbol you push the cdr onto the stack and return the car. Otherwise, you can push the cdr then the car and recurse.

   return a return a return b return a

-> -> (a (b)) -> (b) -> ->

(a ((a (b)) c)) ((a (b)) c) (c) (a) (a) ()

subst-leftmost (HW 8)

I was working on subst-leftmost and my current struggle is when accessing a sublist within the s-list. Currently, it will recurse through that sublist, but then it will also continue to recurse over the cdr of the s-list even when the leftmost has been changed within the car (the sublist). How do I prevent this from occurring and additional modifications being made to the s-list? I am thinking we need to return two things in order to see if the sublist changed anything, but I am not sure how to go about doing this?

[hw8](https://piazza.com/class/k37lfj6dtp3c4?cid=28)

**the students' answer,**

*where students collectively construct a single answer*

[Actions](https://piazza.com/class/k37lfj6dtp3c4?cid=28)

If you want to only sometimes recurse into the cdr, then you'll have to use some sort of syntax like if or cond that doesn't execute all of its conditions. If you want to return more than one thing, the easiest thing to do is just return a list of several things.

slist-leaf-iterator (HW 8)

I've been trying to figure out the slist-leaf-iterator for a while now and there is one part of the instructions that is giving me some confusion. In the instructions is says to use the stack to keep track of cdrs of the pairs who's cars we've already visited. Does this mean that we are intended to keep a list of cars and cdrs in the stack that we turn into a function to get to the next symbol?

**he instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/k37lfj6dtp3c4?cid=29)

"keep a list ... on the stack" is contradictory. I'ts a stack, and you push sublists of the original slist onto it.  Initially, a reference to the entire slist.  After that the stack will contain (at any one time) some unvisited cdrs and at most one car.  If there is a car, it will be on the top of the stack.