Previous questions and answers on Piazza

# Map-by-position

I'm not sure I get what the procedure is supposed to do.

Is it supposed to apply each fn-list to each object in the arg-list? And does it do it in that order?

Or does it apply the corresponding function to the corresponding object?  
I thought that's what it was, but I don't see which elements are corresponding since the example one gives functions that don't apply to the argument it's matched up with.

**the instructors' answer,**

*where instructors collectively construct a single answer*

It is your second case, corresponding.

Look more closely. The procedures and arguments in my example DO correspond. Just as in the example from the beginning of yesterday's class, **list** is not one of the procedures in the list.

[**edit**](https://piazza.com/class/is9cjqgxyh31b?cid=49)·[good answer](https://piazza.com/class/is9cjqgxyh31b?cid=49)0

Updated 5 months ago by Claude Anderson

**followup discussions**

I think I see my mistake now, is "list" not one of the functions?  That would be my problem...

Reply to this followup discussion

Resolved Unresolved



[**Claude Anderson**](https://piazza.com/class/is9cjqgxyh31b?cid=49) [5 months ago](https://piazza.com/class/is9cjqgxyh31b?cid=49)

list just makes a list of the functions

# qsort: odd error 'incorrect argument count in call'

In the partition method I wrote for qsort, I've been getting an odd error that tells me there may be an incorrect argument count in call "(partition pred? (cdr ls) (list (car ls)))". It's the same line of code every time. But that method is defined right under qsort, as having 3 arguments "(define partition  
(lambda (pred? ls pivot)" Oddly enough, if I cut the code out, load it, then put the code back in, it will load perfectly fine without errors and work correctly. However, I'm getting this error when uploading to the server now and have no idea how to fix it. The cases that fail on the server work fine locally, they just give that error on the server.

[hw7](https://piazza.com/class/is9cjqgxyh31b?cid=50)

[**edit**](https://piazza.com/class/is9cjqgxyh31b?cid=50)·[good question](https://piazza.com/class/is9cjqgxyh31b?cid=50)0

**the instructors' answer,**

*where instructors collectively construct a single answer*

There is a link on the Schedule page (Day 1, resources column) called "What if the grading program gives zero points for something that works on your computer?"  Follow it and read the document.

I believe that this document describes your problem and tells you how to fix it.

The built-in procedure that you are redefining is "partition".

# Sort-list-of-symbols: Clarification

So to clarify, we aren't allowed to use any sort of recursion or looping for this, or are we allowed to use recursion with map?

**the instructors' answer,**

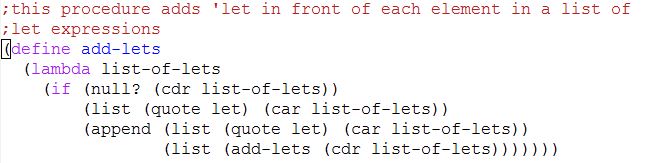
*where instructors collectively construct a single answer*

You may use map.  The function that you are mapping over the list can (I think must) be one that you write, but it cannot be call itself or call another procedure that calls it.

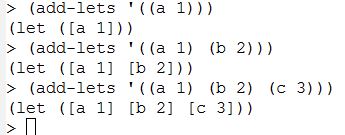
# Let->let\*: getting let to not combine

Im working on the let\*->let problem and i made a helper function that take in the chunk of the passed list that holds the various let expressions (i.e '((a 1)  (b 2) (c 3))) and is supposed to put a let in form of each expression (i.e. create '(let (a 1) (let (b 2) (let (c 3))))) but no matter what combination of cons/list/append I use scheme/emacs is combining my let expressions into one (even though each recursive call i use an individual (quote let)). is there a way to circumvent this?

this is my helper function code:



 and the outputs I get



**the instructors' answer,**

*where instructors collectively construct a single answer*

I believe I see what's going wrong for you here. The initial issue is your function add-lists is using a lambda with what's termed "var-args". If you don't enclose list-of-lets in parentheses, then the function will 1) accept any number of arguments, and 2) collect all its arguments, themselves as a list, in list-of-lets.

To take a small example:

> (define simple-example

(lambda x

(list x x x)))

> (simple-example 'foo)

'((foo) (foo) (foo))

> (simple-example 'foo 'bar 'baz)

'((foo bar baz) (foo bar baz) (foo bar baz))

>

These so-called 'var-args' functions can be useful. For instance

> (+ 1 2)

3

> (+ 1 2 3)

6

We just saw another use above, with list. As a fun exercise, think about how to implement list as a var-args function.

The upshot, I think, is that if you add a set of parens around the function argument, Scheme should instead treat this as an ordinary, one-argument function and I believe will give the behavior you're intending.

BST problem:

So does scheme have INT\_MAX and INT\_MIN?

**the instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/jl863803n0a6tl?cid=42)

As far as I know, the size of integers is limited only by the amount of memory that is available to Scheme.  For example, use the definition of factorial from the day05 folder in Live-in-class.  I tried computing the factorial of 20,00, and it worked just fine.  The answer ts more than 77,000 digits long.  None of the assigned problems require knowing these max and min numbers.  Just use the numbers in the tree (perhaps  with a post-order traversal).

# Questions for assignment 6

I am having trouble finding the questions for assignment 6a. The homework sheet for assignment 6 says the question is exercise 1.3.4 on page 28 in EoPL, but all I can find on page 28 are exercises 1.xx and above, but none in the format of 1.x.x. Furthermore, none of the questions mention currying. Is there something I'm not understanding?

**the students' answer,**

*where students collectively construct a single answer*

[Actions](https://piazza.com/class/jl863803n0a6tl?cid=43)

You'll find the problems in EoPL-1, which is the handout/packet we received in class that had four book pages copied onto each page.

# bt-inorder-list Can I use filter-in?

Can I just make a list of in-order traversal of the whole tree then filter in all the symbols? Or I need to check if the value is a symbol when I make the list?

In this problem, does it mean all interior nodes symbols and all leaves integers? If so it would be easier not to use filter-in.

**the instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/jl863803n0a6tl?cid=46)

* [Flag as Inappropriate](https://piazza.com/class/jl863803n0a6tl?cid=46)

Best:  When you traverse the tree to make the list, don't include the leaf nodes.  Then you don't have to remove them.

# Quicksort Algorithm: Does the runtime have to be strictly N(logN) best case?

The implementation I have in mind behaves like Java Quicksort, but has a longer run time than best-case N(log N).

[hw6](https://piazza.com/class/jl863803n0a6tl?cid=47)

[**edit**](https://piazza.com/class/jl863803n0a6tl?cid=47)·[good question](https://piazza.com/class/jl863803n0a6tl?cid=47)0

Updated 2 months ago by

Eric Tu

**the instructors' answer,**

*where instructors collectively construct a single answer*

I can't think of any reasonable approach that would not be N log N in the best case.  If it is not N log N, is it really quicksort?

Problem 7 - assumptions about data in BST

For problem 7 on homework 6, if we try to insert a new value that already exists in the BST, should we just not insert it? If not, which side should it be inserted on?

**the instructors' answer:**

It is said int the problem that if num is already in bst, result is structurally equivalent to bst, so just don't insert it.

# Making a list out of InOrder traversal

I can't seem to figure out a way to format the result for this function into a singular list. I have it all "in order," but not in a list. Is there a simple way to do this?

**the instructors' answer,**

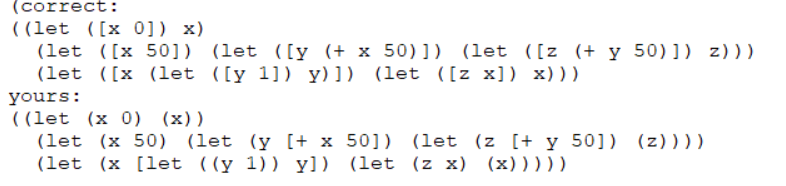
Use a combination of append and list.  I don't think I can say much more without entirely giving it away.

**Followup: I**s it possible to do it without append for efficiency?

**the instructors' answer:**   Probably possible, but not at all easy.

# Assignment 6b: How to add square bracket?

I tried to add brackets to the result but somehow some brackets appear but some disappeared.



**the students' answer:**

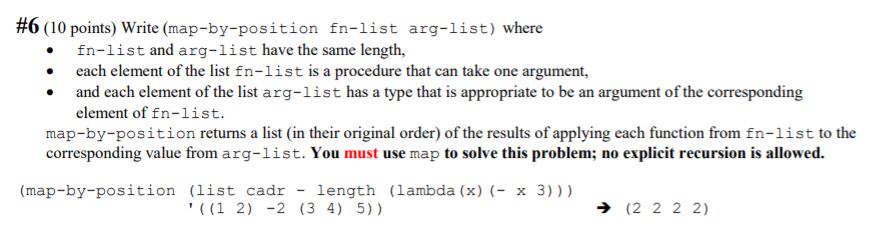
You don't need to add the square brackets by yourself. If your list structure is correct, the parenthesis will be turned into square brackets when needed.

Reply: Got it. Do you know what kind of list structure can cause a square bracket?

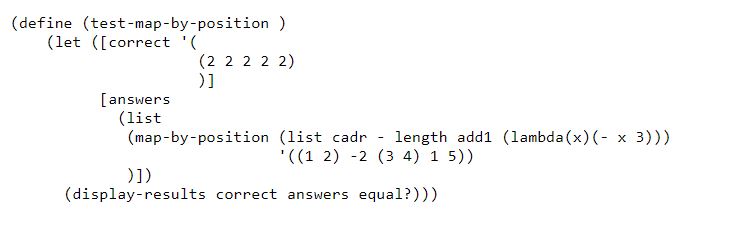
Reply: If you look at your answer, you have square brackets, but they are at the wrong places. This is because your list structure is incorrect and the tests are turning parentheses into brackets where they are not supposed to be turned. The correct list structure should look like the "correct" version above your answer.

# #6 on Assignment 6a

I'm a little confused by this question. The question says that fn-list and arg-list should have the same length but the example case shows it having 5 procedures in the fn-list and 4 arguments in the arg-list



In the example case, it appears to have 6 procedures in the fn-list and only 5 arguments in the arg-list.



**the instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/ke8ite9gsc64w2?cid=62)

Look closely.  I believe that in the example case, there are 4 functions and 4 arguments.

This is all explained near the end of the "map and apply" video.

# Time limit on quicksort?

Is there a time limit on quicksort?

**the instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/ke8ite9gsc64w2?cid=63)

It's a very generous time limit, 100 ms.  The average run of a test case on student code so far is about 0.35 ms.

# #6 on Assignment 6a Recursion

#6 says that we have to use map and no explicit recursion. Does this mean no recursion at all in the solution?

**the students' answer,**

*where students collectively construct a single answer*

[Actions](https://piazza.com/class/ke8ite9gsc64w2?cid=64)

I believe so. If you are using recursion you might be making it a little too complicated in your head. The way that I got started on this problem was to recall what I did for matrix-inverse. The important thing is, map can take more than two arguments.

**~ An instructor (**

**Claude Anderson**

**) endorsed this answer  ~**

**the instructors' answer,**

*where instructors collectively construct a single answer*

Yes.  All of the things you would normally do recursively should be done by using **map**instead.  Thus none of your helper procedures can call themselves.

# qsort should not use list-ref

This is not a requirement, since it wasn't in the specification, but you should try to make your code as efficient as you reasonably can.

A student who was in my office hours used list-ref in his code.  Since list-ref is Θ(N) there is no way that qsort can be N log N even in the best case if you use list-ref.  When doing the partition phase of qsort, use car, cdr, and cons, each of which is constant time.

Letrec restriction

One thing that I've been struggling to grasp well is why, when using letrec, should the exprs be evaluated prior to the vars? For example, why is this operation illegal?

(letrec ( [x (+ y 1)]

[y 1] )

x)

I think you may be confusing the behavior of letrec with let\*. The behavior of letrec can be a bit confusing. In order, it (1) bind all the variables to some undefined value, (2) evaluates the expression of each variable (in an unspecified order), (3) assigns each variable to the result, then (4) evaluates the body. The expression you currently have is trying to evaluate (+ y 1) while y is undefined. Notice that the following expression won't work for the same reason:

(letrec ([y 1] [x (+ y 1)])

x)

The behavior is documented a bit more in the [docs](https://scheme.com/tspl3/further.html).

**the instructors' answer,**

*where instructors collectively construct a single answer*

[Actions](https://piazza.com/class/ke8ite9gsc64w2?cid=69)

The order of evaluation of those initializing expressions in letrec, like in let, is unspecified.  There is a letrec\*.

My suggestion: Only use letrec to define and name local **procedures**, not just any local variables.  Then the lambdas will prevent the bodies from getting executed sooner than yo want.

In the interpreter project, we will assume that letrec in our interpreted language is only used in this restricted way.