MC28 Final Exam

Serial #:

This exam is an open-book, open-notes, individual-work exam. Please write your name only on this page. Do not turn the page until instructed, in order that everyone may have the same time. Then, be sure to look at all problems before deciding which one to do first. Some problems are easier than others, so plan your time accordingly. You have two hours to work.

Write the answer to each problem on the page on which that problem appears. You may use the back of that same sheet if you need additional space. You may also request additional paper.

Name: ______________________________

Problem	Page	Possible	Score
1	2	12	
2	3	12	
3	4	12	
4	5	12	
Total		48	

1. [**12 Points**] Comic strip characters frequently say things like "auggghhh!".

   (a) Write an EBNF grammar that describes all strings that like this one start with "au" and end with "!", and in between have some number of "g" characters followed by the *same* number of "h" characters. Ensure that there is at least one "g" and hence at least one "h". That is, "augghh!" is also in this language, but "aughh!" isn't, and neither is "au!".

   (b) Draw the parse tree for the string "augghh!" given your grammar from the previous sub-problem.

2. [**12 Points**] Consider an extension to Scheme such that instead of writing `(vector-ref v i)` you could just write `(v i)`. Of course in place of the `v`, any expression evaluating to a vector would be legal, and similarly any expression evaluating to an index could be used in place of `i`.

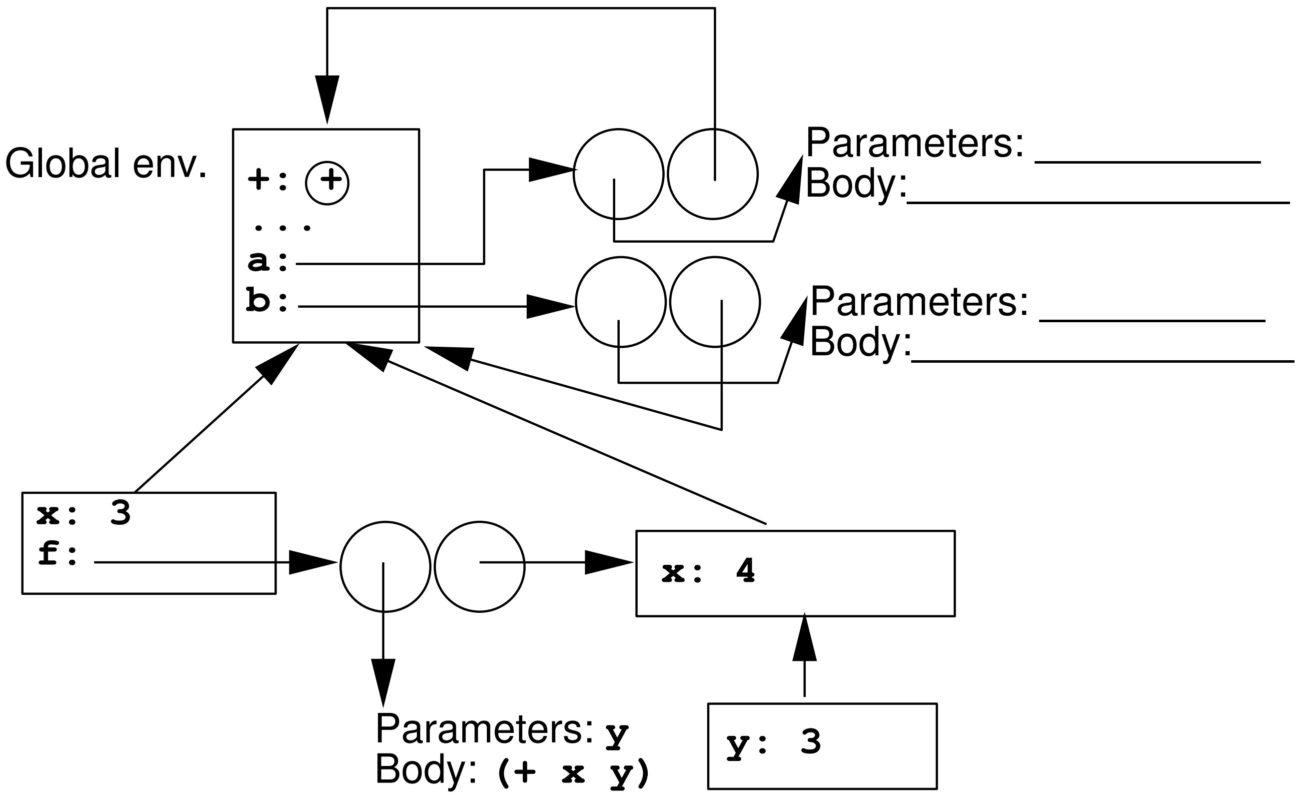
   How would you change the meta-circular evaluator in the textbook (pp. 299–309) to implement this language extension?

3. [**12 Points**] A *collection* is created starting with a single number, and it can have any number of additional numbers added to it. It can report the minimum, maximum, and average of the values in the collection. Here is an example of one in use (the ⇒ arrow is an abbreviation for what would really show up as `;Value:`):

```
(define c (make-collection 3)) ; a collection with only 3 in it so far
(collection-min c) ⇒ 3
(collection-max c) ⇒ 3
(collection-avg c) ⇒ 3
(add-to-collection! 7 c)       ; now has 3 and 7
(collection-min c) ⇒ 3
(collection-max c) ⇒ 7
(collection-avg c) ⇒ 5
(add-to-collection! 2 c)       ; now has 2, 3, and 7
(collection-min c) ⇒ 2
(collection-max c) ⇒ 7
(collection-avg c) ⇒ 4
```

Implement these five operations, representing collections in a way that doesn't require more storage space as additional values are added. That is, the values should *not* simply be added to a list or similar structure, since that would require space proportional to the number of added values.

4. [**12 Points**]



This environment diagram was produce by evaluating the following two definitions and one expression:

```
(define a ____________________)
;Value: a

(define b ____________________)
;Value: b

____________________
;Value: ______
```

No other definitions or expressions have been evaluated, and there are no environment frames or procedure objects other than those show above. The ... in the global environment frame represents only other bindings of standard built-in names, such as the binding of the name * to the multiplication procedure.

Fill in the eight blanks so they are consistent with each other and with the given diagram. (There is more than one correct answer.)