

Name:

CIS 655

Mid-term #2
(Closed book, notes, neighbor)

Sample.

There are seven questions. The first five are worth 10 points each, the sixth worth 20, and the last worth 30. Answer all questions. Be clear, precise, and to the point. For questions that ask if something is possible, if your answer is “yes”, ideally you should provide a simple example to justify your claim.

When grading the first mid-term, I was a bit too liberal, I think, with partial credit for wrong answers. That was part of the reason why the average was rather high. This time, I will be somewhat more strict; so make sure, as it says above, to answer questions precisely and clearly.

When answering questions that ask you to write Lisp/Scheme functions, assume that the language provides only primitive functions; these being *car*, *cdr*, *cons*, *pair?*, *null?*, *eq?*, *+*, *-*, ***, *<*, and *>* where note that *pair?* returns *#t* if its argument is non-atomic and *#f* otherwise; and *null?* returns *#t* if its argument is *nil* (except that *scheme48* expects this to be written as *()*, but you may use *nil*) and *#f* otherwise; you may also assume *cond*, *quote*, and *define* are available. If you prefer, you may use the “design notation” we used in some of our discussions. Also you may always define any extra functions that you need, but please explain your definitions so I understand how they work.

1. Briefly explain the difference between “functions” and “forms”, such as COND in Lisp. Can the Lisp/Scheme user define new forms? If yes, give a simple example; if not, explain why not.

2. In our discussion of the *Core* interpreter, the parse tree played a central role. But we didn’t introduce such a concept when discussing the Lisp interpreter; why was that? Explain briefly.

3. Can a function that receives a list as an argument, return an s-expression that cannot be expressed in the form of a list, as result? Explain briefly. And can a function that receives an s-expression that cannot be expressed as a list as an argument, return a list as result? Explain briefly.

4. Lisp programmers are often not quite sure whether something needs to be quoted. Consider the following advice: “Quote if you are not sure”; is this advice reasonable? How about “avoid quoting if you are not sure”; is that advice reasonable? Explain briefly.

5. What does the following function do? Are there any special cases?

```
(define (strange x) (cons (car x) (cdr x)))
```

Does it behave differently for lists on the one hand versus s-expressions on the other? If there is something wrong with the definition, explain that.

6. We said that the Lisp interpreter “looks up the value of variables on the association list”. How exactly does it do this? Can this action be expressed as a Lisp function? I.e., can you write a Lisp function `getVal` that takes two arguments, `x`, `aL`, and looks up the value of `x` on `aL`, and returns that as its result? If yes, show how you would do this; if not, explain why you cannot do this.

7. Write a Lisp function *biggerList* that takes two lists *L1* and *L2* of numbers as arguments, and returns the list *L1* as the result if the largest value in *L1* is greater than the largest value in *L2*, and otherwise returns the list *L2* as the result. If you believe such a function cannot be written in Lisp/Scheme, explain what the problem is.