CS 61A Midterm #2 — March 10, 2004

Your name .

login: cs61a-

Discussion section number

TA's name

This exam is worth 40 points, or about 13% of your total course grade. The exam contains six substantive questions, plus the following:

Question 0 (1 point): Fill out this front page correctly and put your name and login correctly at the top of each of the following pages.

This booklet contains eight numbered pages including the cover page. Put all answers on these pages, please; don't hand in stray pieces of paper. This is an open book exam.

When writing procedures, don't put in error checks. Assume that you will be given arguments of the correct type.

Our expectation is that many of you will not complete one or two of these questions. If you find one question especially difficult, leave it for later; start with the ones you find easier.

READ AND SIGN THIS:

I certify that my answers to this exam are all my own work, and that I have not discussed the exam questions or answers with anyone prior to taking this exam.

If I am taking this exam early, I certify that I shall not discuss the exam questions or answers with anyone until after the scheduled exam time.

	1	
0		/1
1		
2		
3		
4		
5		
6		
total		/40

Question 1 (6 points):

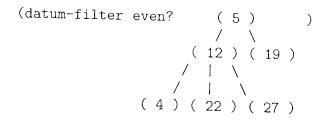
What will Scheme print in response to the following expressions? If an expression produces an error message, you may just write "error"; you don't have to provide the exact text of the message. Also, draw a box and pointer diagram for the value produced by each expression.

Your name	
_	login cs61a

Question 2 (7 points):

Write the function datum-filter which, given a predicate and a tree, returns a list of all the datums in the tree that satisfy the predicate (in any order). We are using general trees (trees that can have any number of children) as defined in lecture. We are not using binary trees. The function should return the empty list for any tree in which no datums satisfy the predicate. You may use helper procedures.

For example:



returns the list $(12\ 4\ 22)$ in any order.

(define (datum-filter pred tree)

Question 3 (7 points): Many people write cond expressions like this:

```
(cond ((= x 5) 100) (else 200))
```

in which there are only two cond clauses, the second of which is an else clause. This can be written as the more aesthetically pleasing

```
(if (= x 5) 100 200)
```

We want a procedure, simplify-cond, that takes a valid Scheme expression as its argument, and replaces all two-clause-with-else cond expressions by an equivalent if expression. Bear in mind that cond expressions can be nested:

Here is an attempt to write simplify-cond, but it has two bugs. Your job is to find and fix the bugs. (The helper procedure two-clauses-cond-exp? is correct!)

```
(define (simplify-cond exp)
    (if (two-clauses-cond-exp? exp)
        (list 'if (car (cadr exp)) (cadr (cadr exp)) (cadr (caddr exp)))

(define (two-clauses-cond-exp? exp)
    (and (pair? exp)
        (eq? (car exp) 'cond)
        (= (length exp) 3)
        (eq? (car (caddr exp)) 'else)))
```

Your name	login cs61a-
Question 4 (7 points):	
Here is the constructor for a new ADT designe	ed to store student information.
(define (make-student name age ad) (list name 'age age (list 'id id)))	
(a) Define the selectors for this student data ty	ре:
(define (name student))
(define (age student))
(define (id student))
(b) Write get-student, which takes a list of student who matches that ID. If no student in t #f. You may not define any helpers other than	the list matches the ID, the function returns the selectors you wrote in part (a).
(define (get-student list-of-students st	nudent-id)

Question 5 (6 points): Here is the code for the dyadic version of the sentence procedure using conventional style:

We're going to re-design this procedure using data directed programming. We'll see type signatures such as (word word) or (sentence word) corresponding to the two arguments to sentence.

Rather than adding explicit type tags to words and sentences, we'll use the actual Scheme types as implicit tags, by rewriting the selectors for tagged data. (This relies on the fact that the explicit types we've used all end up as improper lists such as (rational 3 . 4), so sentence? is false for them.)

Your new version of sentence should behave exactly like the one defined above. Here are some examples:

```
(sentence 'george 'harrison) => (george harrison)
(sentence 'angus '(young)) => (angus young)
(sentence '(eric) '(clapton)) => (eric clapton)
```

This question continues on the next page!

Your name.	login cs61a

Question 5 continued:

(a) Make calls to put to prepare the table:

(b) Write the new sentence procedure in the Data Directed Programming style. You may use apply-generic from the book if you find it useful.

Question 6 (6 points):

Here is the eval-1 code for scheme-1 (before any changes you made in lab or homework):

Suppose we moved the cond clause beginning with pair? (the one that handles function calls) to the beginning of the cond, making it the first clause:

With the above changes, show what scheme-1 would print given the following inputs. If the result is an STk error, just write "ERROR":

```
3 =>
+ (if #t 1 2) =>
((lambda (x) (* x x)) 3) =>
(+ 3 4) =>
(lambda (x) (* x x)) =>
```