CS 60A Final — May 16, 1992

Your name ____

login c60a–_______

Discussion section number _______

TA's name ____

This exam is worth 30 points, or 18.75% of your total course grade. The exam contains six questions.

This booklet contains eleven numbered pages (both sides of six sheets) 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.

1	/5
2	/5
3	/5
4	/5
5	/5
6	/5
total	/30

Question 1 (5 points):

Every integer greater than 1 has a *unique prime factorization*. That is, there is exactly one way that each integer can be expressed as a product of prime numbers. For example, 12 is $2 \times 2 \times 3$. 13 is just 13. 14 is 2×7. 15 is 3×5. (Notice that the same prime can be used more than once, as in the case of 12.)

You are given the streams `integers` (containing all the positive integers) and `primes` (containing all the prime numbers). (You need not reproduce the computations in the book that generate these streams.)

(a) Write the function `factors` that takes an integer as its argument and returns a list (not a stream) containing its prime factorization. For example:

```
> (factors 12)
(2 2 3)
> (factors 13)
(13)
```

(b) Construct the stream `factor-stream` that contains the prime factorizations of all the positive integers.

Your name ____ login c60a–_______

Question 2 (5 points):

We're going to create an abstract data type for times of day (hours and minutes). To make things simpler, we'll use 24-hour times, in which 3pm is hour 15. You are going to implement this abstract data type using two different internal representations.

The constructor for times of day is called `time`. It takes two arguments: the hours and the minutes. So 4:12am would be `(time 4 12)`.

We also want the following three operators for times:

`(hour t)` returns the hour portion of the given time.

`(minute t)` returns the minute portion of the given time.

`(hour? t)` returns `#t` (true) if the time `t` is an exact hour time (like 6:00); it returns `#f` (false) otherwise.

(a) Implement `time`, `hour`, `minute`, and `hour?` using an internal representation in which a time is represented as a list of two numbers. That is, 4:12am should be represented internally as the list `(4 12)`.

This question continues on the following page.

Question 2 continued:

(b) Implement `time`, `hour`, `minute`, and `hour?` using an internal representation in which a time is represented as an integer, namely, the number of minutes since midnight. That is, 4:12am should be represented internally as the number `252` ($4 \times 60 + 12$).

Your name ______________________________________ login c60a–_______

Question 3 (5 points):

King Arthur had a problem. His daughter, Glissanda, was interested in marrying. But she had one requirement for a husband: he must love mathematics. Arthur was confused about how to find such a husband. After all, the Knights of the Round Table (the best men in the land) were outdoor types who went around slaying dragons. He couldn't remember any of them ever mentioning mathematics.

Glissanda suggested that he administer a mathematical test to the Knights. She explained, "Suppose 24 knights came to a meeting of the Round Table. And suppose the 24 chairs were numbered in order, from 1 to 24. In order to choose my husband, you draw your sword, point to the knight in the first chair, and say, 'You live.' Then point to the knight in chair number 2, say 'You die,' and chop off his head. To the third knight you say, 'You live,' to the fourth, 'you die,' and so on, chopping off the head of every other living knight until just one is left. That's the one I'll marry."

"That's it?" asked Arthur, horrified. "You expect me to kill all my knights but one? Is this what you call mathematics?"

"Oh, father," Glissanda said, "I wouldn't expect you to actually kill anyone. It's just a problem. The knight of my dreams will be able to figure out which chair to choose no matter how many knights show up for the test."

[Adapted from *Math for Smarty Pants* by Marilyn Burns. Illustrations by Martha Weston. Copyright © 1982 by the Yolla Bolly Press.]

We want to be able to simulate this process of elimination for different numbers of knights. To this end you'll define a `knight` object class.

A `knight` is created with one instantiation variable, the chair number in which he is seated. Each `knight` object must keep track of its (still living) neighbors to the left and to the right. A `knight` accepts the following messages:

This question continues on the following page.

Question 3 continued:

`Set-left-neighbor!` takes a knight as argument and sets the recipient knight's left neighbor to the argument knight.

`Set-right-neighbor!` takes a knight as argument and sets the recipient knight's right neighbor to the argument knight.

`Skip` (with no arguments) tells the recipient that he's been skipped. The recipient should send a `die` message to his left neighbor, unless he is the only knight left alive, in which case the method should return his number as the winning knight.

`Die` (with no arguments) tells the recipient that he's been killed. The recipient should remove himself from the circle (by telling his neighbors to adjust their neighbors); print a message like `knight 12 dies`; and send a `skip` message to his left neighbor.

Example: To run a simulation with three knights, you would do this:

```
(define knight-1 (instantiate knight 1))
(define knight-2 (instantiate knight 2))
(define knight-3 (instantiate knight 3))
(ask knight-1 'set-right-neighbor! knight-2)
(ask knight-2 'set-right-neighbor! knight-3)
(ask knight-3 'set-right-neighbor! knight-1)
(ask knight-1 'set-left-neighbor! knight-3)
(ask knight-2 'set-left-neighbor! knight-1)
(ask knight-3 'set-left-neighbor! knight-2)
(ask knight-1 'skip)
```

Your name ____ login c60a–________

Question 4 (5 points):

Write the function `cxr-function`. This function must take as its argument a symbol such as `CDDDADDAADAR` and returns the function that should have that name! (Note, this is not the same as the bonus problem we gave earlier. The bonus problem went from function to name. We want to go from name to function. This is much easier.)

You may assume that the argument is a symbol that starts with `C`, ends with `R`, and has some combination of `A`s and `D`s in between. You must return a function of one argument that returns the result of the corresponding composition of `car`s and `cdr`s of that argument. The symbol may be of any length.

Examples:

```
> ((cxr-function 'cadr) '(a b c d e))
B
> (define func (cxr-function 'cddaddadr))
FUNC
> (func '((1 2 3) (x y (red orange yellow green blue) z) (a b c)))
(YELLOW GREEN BLUE)
```

Question 5 (5 points):

(a) Write a logic program (one or more rules) to implement the `substring` relation. It takes two lists and it succeeds if one is a substring of the other. For example, the query

```
(substring (to hold your) (i want to hold your hand))
```

should succeed, but the query

```
(substring (to hold hand) (i want to hold your hand))
```

should fail. (Hint: One possible solution uses the `append` rules; another possible solution doesn't require `append`.)

(b) We don't expect you to be able to predict exactly what'll happen, but assuming there are no strange bugs, what might you reasonably expect if you tried the query

```
(substring ?x (a b c))
```

____ "Done" with no matches reported.

____ "Done" with one match reported.

____ "Done" with more than one correct match reported.

____ An infinite loop.

(c) What might you reasonably expect from the query

```
(substring (a b c) ?x)
```

____ "Done" with no matches reported.

____ "Done" with one match reported.

____ "Done" with more than one correct match reported.

____ An infinite loop.

Your name ____ login c60a–_______

Question 6 (5 points):

Alyssa P. Hacker has noticed that many 60A students lose points in midterm 3 by leaving out the empty frames that you get from invoking a procedure with no arguments. She asks all her friends how she can teach the students better so they won't make this mistake.

Ben Bitdiddle says, "Instead of teaching the students better, let's just modify the metacircular evaluator so that it doesn't create frames that would be empty. Then the students will be right."

Modify the metacircular evaluator to implement Ben's suggestion.

(a) Is this primarily a change to `eval` or to `apply`?

(b) What specific procedure(s) will you change?

(c) Make the changes on the following pages.

(d) Lem E. Tweakit notices that this change is not such a great idea. He thinks the modified metacircular evaluator will interpret certain Scheme programs incorrectly.

Under what circumstances would this modification to the evaluator change the meaning of Scheme programs run using the metacircular evaluator? (That is, what kind of program will produce different results using the modified version than using the original one?)

This question continues on the following page.

Question 6 continued:

```
(define (eval exp env)
  (cond ((self-evaluating? exp) exp)
        ((quoted? exp) (text-of-quotation exp))
        ((variable? exp) (lookup-variable-value exp env))
        ((definition? exp) (eval-definition exp env))
        ((assignment? exp) (eval-assignment exp env))
        ((lambda? exp) (make-procedure exp env))
        ((conditional? exp) (eval-cond (clauses exp) env))
        ((application? exp)
         (apply (eval (operator exp) env)
                (list-of-values (operands exp) env)))
        (else (error "Unknown expression type -- EVAL" exp))))
```

```
(define (apply procedure arguments)
  (cond ((primitive-procedure? procedure)
         (apply-primitive-procedure procedure arguments))
        ((compound-procedure? procedure)
         (eval-sequence (procedure-body procedure)
                        (extend-environment
                         (parameters procedure)
                         arguments
                         (procedure-environment procedure))))
        (else (error "Unknown procedure type -- APPLY" procedure))))
```

```
(define (list-of-values exps env)
  (cond ((no-operands? exps) '())
        (else (cons (eval (first-operand exps) env)
                    (list-of-values (rest-operands exps)
                                    env)))))
```

```
(define (eval-sequence exps env)
  (cond ((last-exp? exps) (eval (first-exp exps) env))
        (else (eval (first-exp exps) env)
              (eval-sequence (rest-exps exps) env))))
```

This question continues on the following page.

Your name ____ login c60a–_______

Question 6 continued:

```
(define (extend-environment variables values base-env)
  (adjoin-frame (make-frame variables values) base-env))



(define (adjoin-frame frame env) (cons frame env))



(define (make-frame variables values)
  (cond ((and (null? variables) (null? values)) '())
        ((null? variables)
         (error "Too many values supplied" values))
        ((null? values)
         (error "Too few values supplied" variables))
        (else
         (cons (make-binding (car variables) (car values))
               (make-frame (cdr variables) (cdr values))))))



(define (make-binding variable value)
  (cons variable value))



(define (eval-definition exp env)
  (define-variable! (definition-variable exp)
                    (eval (definition-value exp) env)
                    env)
  (definition-variable exp))



(define (make-procedure lambda-exp env)
       (list 'procedure lambda-exp env))
```