University of California, Berkeley – College of Engineering

Department of Electrical Engineering and Computer Sciences

Summer 2002 Instructor: Kurt Meinz 2002-07-26

CS 61A Midterm #2

Personal Information

First and Last ***Name***	
Your ***Login***	cs61a-__ __ *(legibly!)*
The ***First Letter*** *of Your Login (please circle)*	a b c d e f g h i j k l m n o p q r s t u v w x y z
The ***Second Letter*** *of your login (please circle)*	a b c d e f g h i j k l m n o p q r s t u v w x y z
Lab Section Time, TA, & Location ***You Attend***	
Discussion Section Time, TA, & Location ***You Attend***	
"All the work is my own. I had no prior knowledge of the exam contents nor will I share the contents with others in CS61a who have not taken it yet."	***(please sign)***

Instructions

- Partial credit may be given for incomplete / wrong answers, so please write down as much of the solution as you can.
- Feel free to use any Scheme function that was described in lecture or sections of the textbook we have read without defining it yourself. Do not use functions or constructs that we have not yet covered. Unless specifically prohibited, you are allowed to use helper functions on any problem.
- Please use "true" instead of #t , and "false" instead of #f. We have found that handwritten #t and #f unfortunately look too much alike.
- Please write legibly! If we can't read it, we won't grade it!

Grading Results

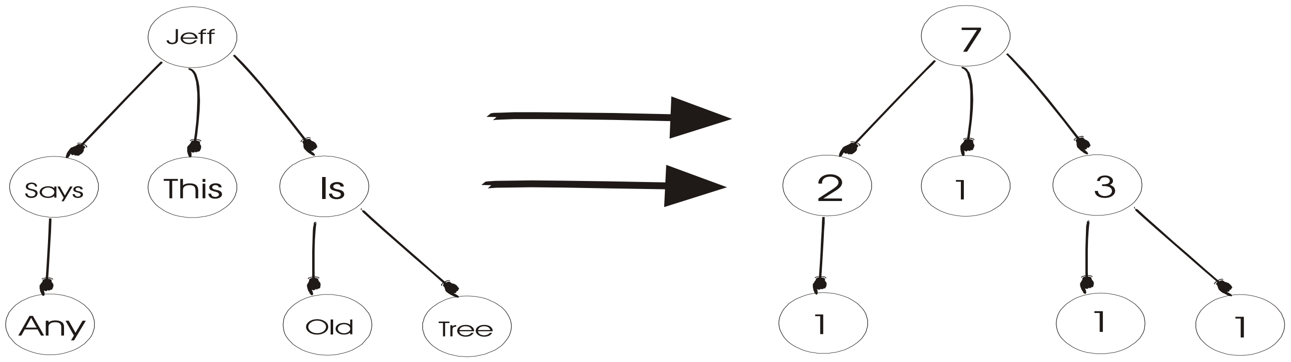
Question	*Max. Points*	*Points Earned*
1	**6**	
2	**10**	
3a	**4**	
3b	**6**	
3c	**2**	
4	**12**	
Total	**40**	

Name: ___________________________ Login: _____________________

Question 1: Tree Recursion [6 Points]

Please write a procedure named ‘count-nodes’ that takes a tree as an argument and returns a new tree with the same structure but with each datum replaced with the total number of nodes in that subtree. Please use the tree abstraction given in lecture (‘make-node’, ‘children’ and ‘datum’).

For example,



```
(define (count-nodes node)
```

Name: ______________________ **Login:**_____________________

Question 2: Data Directed Programming **[10 Points]**

Jane and Erwin want to gossip behind Kurt’s back [“He wears ugly shirts,” says Erwin.] but they are afraid Kurt will intercept their email messages. “I’ve got it!” Erwin says, “Let’s encrypt our communications.” Here’s how they decided to do it: The encoder takes a single non-numeric word and encrypts it using an encryption procedure (named ‘encrypt-0’ through ‘encrypt-9’). They then take the encrypted word, prepend the number of the encryption method, and either send this encrypted word or (for more security) encrypt it again with another method, and so on.

For example, to double-encrypt ‘shirts’, Jane would do this:

```
(word 3 (encrypt-3 (word 7 (encrypt-7 'shirts))))
```

To decrypt words, Erwin wrote this (correct) procedure:

```
(define (decrypt message)
 (cond ((not (number? (first message))) message)
       ((= 0 (first message)) (decrypt (decrypt-0 (bf message))))
       ((= 1 (first message)) (decrypt (decrypt-1 (bf message))))
        …))
```

Of course, Erwin’s procedure would have been better written using DDP. Please re-write ‘decrypt’ so that it uses DDP. You may assume that the decryption procs are named ‘decrypt-0’ through ‘decrypt-9’. However, we want you to experiment with using storage structures other than tables, so solutions that use a table to store the decryption procs will earn a maximum of ¾ of the points. [Hint: think linearly.] Solutions will be graded, in part, on elegance.

```
(define (decrypt message)
```

Name: ______________________ **Login:**______________________

Question 3: OOP Environments [12 Points Total]

Ilya has grown tired of drawing environment diagrams. He says, "Why draw them when I could just define a frame class that will simulate environments for me?" Such a class would have these properties:

1. Frames should provide a mechanism for binding, re-binding, and looking-up variable values.
2. Frames should be able to extend other frames.

In this question, we would like you to provide an OOP implementation for these frames. Since there are many possible solutions to this question, grading will be partly based on the elegance and simplicity of your solution.

We are leaving the exact details of the specification open to encourage you to think critically about what *Exactly* a frame class should consist of. Thus, variable names, method names, etc. are up to you.

Part I: Define a basic frame class that can do environment extension. In particular, you must provide a way to create a global frame as well as a mechanism to extend previously-instantiated frames. Don't worry about variable bindings yet, but do be cognizant that you will extend this class to handle variables in Part II.

(Part II is on the next page.)

```
(define-class (basic-frame
```

Name: ________________________ **Login:**_______________________

Question 3: OOP Environments (continued)

Part II: Define a more advanced frame class that inherits frame-extension abilities of the frames from Part I but also does variable binding, re-binding, and lookup. You may assume that the input to your classes will be valid. (E.g. variable definition (with 'define') will always take place before assignment (with 'set!'). [Hint: Don't forget to search the enclosing environment when necessary.]

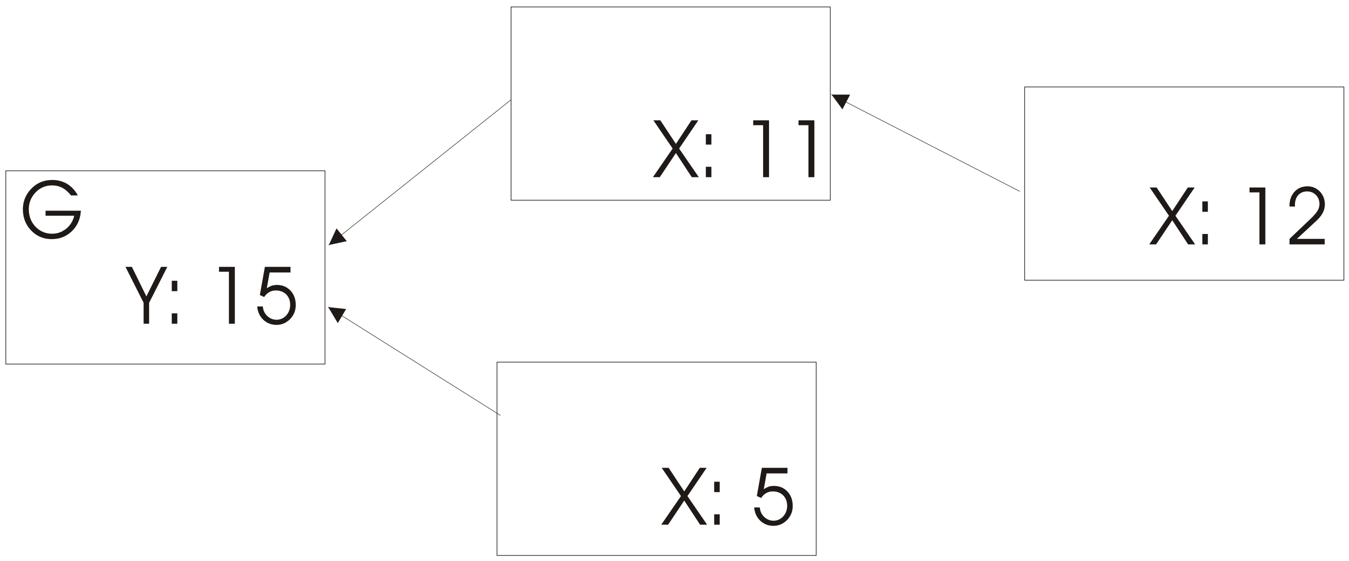
You may want to use local tables to store bindings. ['make-table', 'put <table> <coordinates> <val>', 'get <table> <coordinates>'.]

```
(define-class (var-frame
```

Name: ________________________ **Login:**_______________________

Question 3: OOP Environments (continued)

Part III: If you did your frame classes correctly, you should now be able to provide a series of expressions that will create the following frame structure:



Please do so here:

Name: ________________________ **Login:**_______________________

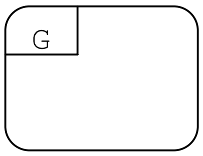
Question 4: Environments Ouch! **[12 Points]**

Consider this scheme expression:

```
((lambda (erwin)
  ((lambda (jeff) (jeff jeff erwin))
   (lambda (greg ilya)
     (if (= ilya 1)
         1
         (* (greg greg (- ilya 1)) ilya)))))
 3)
```

Part I: Assuming that this expression is evaluated with reference to the top-level, global frame (G0), draw an environment diagram that captures all of frames and procedures created by the evaluation. Please label your frames starting at F1 where each number corresponds to the order of creation of the frame. Don't forget to include parent pointers, googly-eyes, and bindings! (You don't have to fill in the 'body' section of the googly-eyes, but you should fill in the 'arg' section.)

Don't forget to do part II at bottom!



Part II: Describe, in no more than 6 words, what this expression calculates when evaluated.

____