Your name ____________________________

login: cs61a-_____

Discussion section number _______

Circle your TA's name
Chris Cheng
Chris Karlof
David Schultz
Donald Chai
Erwin Vedar
Ryan Stejskal
Seema Moorjani
Todd Segal
Yaping Li

The student sitting to my right is ____________________________

The student sitting to my left is ____________________________

This exam is worth 50 points, or about 17% of your total course grade. It includes two parts: The individual exam (this part) is worth 44 points, and the group exam is worth 6 points. The individual 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 six 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.

<table>
<thead>
<tr>
<th>READ AND SIGN THIS:</th>
<th>0</th>
<th>/1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>/12</td>
</tr>
<tr>
<td></td>
<td>2-3</td>
<td>/7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>/8</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>/8</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>/8</td>
</tr>
<tr>
<td></td>
<td>total</td>
<td>/44</td>
</tr>
</tbody>
</table>

1
Question 1 — I’m drawing a blank... (12 points):

(a) Fill in the simplest possible answer in the blank so that the expression produces the result.

\[\lambda(x) (x \ x \ x)\]

usa

(b) Given the following two definitions, fill in zero or more parentheses only (PO) in each blank to complete the scheme expression and show the numeric return value on the right.

\[
\text{(define (double x) (* 2 x))}
\]

\[
\text{(define olympics}
\lambda(x)
\lambda(f)
\text{(f (f x)))}
\]

\[
> \quad \text{olympics} \quad \text{olympics} \quad 5 \quad \text{double} \quad \text{double} \quad \Rightarrow \quad \text{must be a number}
\]

PO PO PO PO PO PO

(c) Fill in the result. If the expression produces an error, just say “error”; if it returns a procedure, just say “procedure”; if it causes an infinite loop, just say “loop”.

\[
> \text{(let ((y '(gold)))}
\quad \text{(cons (append y (list y)) y))}
\]

(d) Fill in the result. If the expression produces an error, just say “error”; if it returns a procedure, just say “procedure”; if it causes an infinite loop, just say “loop”.

\[
> \text{(define (bronze n)}
\quad \text{(define (iter count i)}
\quad \text{(if (<= i 1)}
\quad \text{count}
\quad \text{(iter (+ count 1) (- i 2)))}
\quad \text{(iter 0 n))}
\]

\[
> \text{(bronze 5000)}
\]

(e) Fill in the result. If the expression produces an error, just say “error”; if it returns a procedure, just say “procedure”; if it causes an infinite loop, just say “loop”. You may assume pigl is defined exactly as it is in the notes.

\[
> \text{(every pigl (se '() 'class '(beats) (word 61 'a)))}
\]
Question 2 — Eve Ensler was discussing “Big-O” notation... (3 points):

This question concerns the following functions.

(define (f x) (if (< x 0) 1 (+ (f (- x 2)))
(define (g y) (if (< y 100) 0 (+ (g y)
(define (h z) (if (< z 0) 0 (+ (h (- z 2)))

Circle (T)true or (F)alse:

T F h generates an iterative process (i.e., uses Θ(1) space).
T F g generates an iterative process (i.e., uses Θ(1) space).
T F f is Θ(x).
T F h is Θ(z^2).
T F f and g have the same order of growth.
T F g and h have the same order of growth.

Question 3 — Waiter, please take my order of evaluation... (4 points):

This question concerns the following functions.

(define (square x) (* x x))

(define (foo x y) (* x y y))

(define (bar x y) x)

(a) How many times is * called for (foo (square (* 1 1)) (square (* 1 1)))

... under normal order? ___________

... under applicative order? ___________

(b) How many times is * called for (bar (square (* 1 1)) (square (* 1 1)))

... under normal order? ___________

... under applicative order? ___________
Question 4 — If you can’t be an athlete, be an athletic supporter... (8 points):

We designed an abstract data type, athlete, to represent Olympic athletes. An athlete contains 3 pieces of information: name (a word); country (a word); and events (a sentence containing the names of all events the athlete participates in). Here is the constructor:

(define (make-athlete name country events)  
  (append (list name country) (cons events '())))

(a) Write the appropriate selectors for the athlete ADT.

(b) We have created a list of athletes as follows:

(define all-athletes  
  (list  
    (make-athlete 'Picabo-Street 'USA (se 'slalom 'downhill))  
    (make-athlete 'Todd 'USA (se 'slalom 'super-monkey-ball))  
    (make-athlete 'Johnny-Moseley 'USA (se 'moguls 'video-games))  
    ... more athletes ... ))

We wrote get-participants that takes an event and list of athletes as input and returns a sentence containing the names of all athletes who participate in the events. E.g.,

> (get-participants 'slalom all-athletes)  \Rightarrow  (Picabo-Street Todd)

Complete the definition of get-participants. Respect the Data Abstraction!

(define (get-participants event list-of-athletes)

  (cond ((________________________________________ list-of-athletes) '())
    (((member? event (__________ (__________ list-of-athletes)))
      (__________ (__________ (__________ list-of-athletes))
      (get-participants event (____ list-of-athletes))))
    (else (get-participants event (__________ list-of-athletes)))))

4
Your name ___________________________ login cs61a--

Question 5 — I can’t talk right now, my head is in a vice... (8 points):

This question concerns the twenty-one project (the version without jokers).

The CS61A staff is going to Las Vegas, and we plan to play a lot of twenty-one! One of the questions in the project asked you to create your own strategy. We would like to figure out which of these strategies is the best, so we can use it in Vegas and win billions.

The procedure best takes an integer n and a non-null list of strategies as arguments. best will play twenty-one with each strategy n times and return the strategy that had the best results—in the case of a tie, any winner may be returned.

Fill in the blanks to complete the definition of best. Remember, you can assume that the list of strategies is not empty. Do not create any new procedures!

(define (best n list-of-strats)

  (define (helper best-strat best-result strats-left)

    (if (__________________________________________)

        __________________________________________

        (let ((result __________________________________))

          (if (< _________________________________________)

              __________________________________________

              __________________________________________)

          )

      )

    (helper ________________________________________

  )

Copyright © 2001 United Feature Syndicate, Inc.
Question 6 — Dial 1-800-call-all and save a buck or two... (8 points):

Louis Reasoner is trying to write a function call-all that takes a list of single-argument functions and returns a new unary function. The function returned by call-all should call each of the function in the list with its input and return a list of the results. E.g.,

```scheme
> (define (identity x) x)
> (define (square x) (* x x))
> (define (cube x) (* x x x))
> (define (powers call-all (list identity square cube)))
> (powers 2)
(2 4 8)
> (define pigl-and-reverse (call-all (list pigl reverse)))
> (pigl-and-reverse 'class)
(assclay ssalc)
> (define no-results (call-all '()))
> (no-results 100)
()```

Louis’ first attempt at writing call-all looks like:

```scheme
(define (call-all L)
  (if (null? L)
      '()
    (lambda (x)
      (cons ((car L) x)
            (call-all (cdr L))))))
```

(a) Complete the sentence below to describe what Louis’ program returns versus what it is supposed to return (according to the specifications). If the expression produces an error, just say “error”; if it returns a procedure, just say “procedure”; if it causes an infinite loop, just say “loop”. “A call to ((call-all (list square 1+)) 2)

returns ____________________________________

but should return ____________________________________.”

(b) Make as few changes as necessary to fix the procedure so that it works as advertised on all inputs. You will probably not need to change all the lines.

Change line ___ to ____________________________________

Change line ___ to ____________________________________

Change line ___ to ____________________________________

Change line ___ to ____________________________________

Change line ___ to ____________________________________