Mid-Term Exam 2

CS 5510, Fall 2005

(take-home, but Fall 2010 exam will be in-class)

Name: ________________________________

Start time: ____________________________
End time: ______________________________

Instructions: You have ninety minutes to complete this open-book, open-note, closed-computer, take-home exam. Please write you start and finish times above, and write all answers in the provided space, plus the back of the exam if necessary. **The Fall 2010 Mid-Term 2 exam will be in-class for 60 minutes.**

1) Which of the following produce different results in a eager language and a lazy language? Both produce the same result if they both produce the same number or they both produce a procedure (even if the procedure doesn’t behave exactly the same when applied), but they can differ in errors reported.

a) `{fun {y} 12} {1 2}`

b) `{fun {x} `{fun {y} 12} {1 2}}`

c) `{+ 1 `{fun {y} 12}}`

d) `{+ 1 `{fun {x} `{+ 1 13}} `{+ 1 `{fun {z} 12}}}`

e) `{+ 1 `{fun {x} `{+ x 13}} `{+ 1 `{fun {z} 12}}}`
2) The following web servlet implementation (main handler plus helper function) uses `web-read`, which takes only a prompt and uses `let/cc` internally to obtain a continuation. Convert the servlet (both functions) to instead use `web-read/k`, which takes a prompt and an explicit continuation procedure (and does not use `let/cc` internally). You should assume that the `correct-password?` function requires no interaction with the user. The Fall 2010 version of this question will be more difficult.

```scheme
(define (pw-handler base args)
  (get-pw (web-read "Name")))

(define (get-pw name)
  (local [(define pw (web-read "Password"))]
    (if (correct-password? name pw)
        (format "Hello, "a" name)
        (get-pw name))))
```
3) Given the following expression:

\[
\{\text{fun } \{x\} \{x \ x\}\} \\
\{\text{fun } \{y\} \{12\}\}
\]

Describe a trace of the evaluation in terms of arguments to \texttt{interp} and \texttt{continue} functions for every call of each. (There will be 7 calls to \texttt{interp} and 5 calls to \texttt{continue}.) The \texttt{interp} function takes three arguments — an expression, a substitution cache, and a continuation — so show all three for each \texttt{interp} call. The \texttt{continue} function takes two arguments — a value and a continuation — so show both for each \texttt{continue} call. Represent continuations using records. \textbf{The Fall 2010 version of this question will involve the continuation-passing interpreter of HW 10 instead of the interp-continue interpreter.}
4) (Extra credit for Fall 2010, since it’s based on lecture instead of homework.) Suppose a garbage-collected interpreter uses the following three kinds of records:

- Tag 1: a record containing two pointers
- Tag 2: a record containing one pointer and one integer
- Tag 3: a record containing one integer

The interpreter has one register, which always contains a pointer, and a memory pool of size 22. The allocator/collector is a two-space copying collector, so each space is of size 11. Records are allocated consecutively in to-space, starting from the first memory location, 0.

The following is a snapshot of memory just before a collection where all memory has been allocated:

- Register: 8
- To space: 1 3 8 3 0 2 3 7 2 0 8

What are the values in the register and the new to-space (which is also addressed starting from 0) after collection? Assume that unallocated memory in to-space contains 0.

- Register:

- To space:
Answers

1) a and d.

2) (define (pw-handler base args)
    (web-read/k "Name" get-pw))

    (define (get-pw name)
        (web-read/k "Password"
            (lambda (pw)
                (if (correct-password? name pw)
                    (format "Hello, ~a" name)
                    (get-pw name)))))

3)

interp expr = \{fun \{x\} \{x\}\} \{fun \{y\} 12\}
subs = (mtSub)
k = (mtK)

interp expr = \{fun \{x\} \{x\}\}
subs = (mtSub)
k = (appArgK \{fun \{y\} 12\} (mtSub) (mtK))

cont val = (closureV 'x \{x\}) = v_1
k = (appArgK \{fun \{y\} 12\} (mtSub) (mtK))

interp expr = \{fun \{y\} 12\}
subs = (mtSub)
k = (doAppK v_1 (mtK))

cont val = (closureV 'y 12) = v_2
k = (doAppK v_1 (mtK))

interp expr = \{x\}
ds = (aSub 'x v_2 (mtSub)) = ds_1
k = (mtK)

interp expr = x
ds = ds_1
k = (appArgk x ds_1 (mtK))

cont val = v_2
k = (appArgK x ds_1 (mtK))
\[ k = (\text{doAppK} \ v_2 \ (\text{mtK})) \]

\[
\begin{align*}
\text{interp expr} &= \boxed{12} \\
\text{ds} &= (\text{aSub} \ 'y \ v_2 \ (\text{mtSub})) \\
k &= (\text{mtK})
\end{align*}
\]

\[
\begin{align*}
\text{cont val} &= (\text{numV} \ 12) \\
k &= (\text{mtK})
\end{align*}
\]

4) Register: 0, To space: 2 3 8 1 6 0 3 0 0 0 0