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 2009 Mid-Term 2 exam will be in-class.**

1) Which of the following produce different results in a call-by-value language and a call-by-name 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).

   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 2009 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:

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

Describe a trace of the evaluation in terms of arguments to \text{interp} and \text{continue} functions for every call of each. (There will be 7 calls to \text{interp} and 5 calls to \text{continue}.) The \text{interp} function takes three arguments — an expression, a substitution cache, and a continuation — so show all three for each \text{interp} call. The \text{continue} function takes two arguments — a value and a continuation — so show both for each \text{continue} call. Represent continuations using records. The Fall 2009 version of this question will involve the store-passing interpreter of HW6 instead of the \text{interp–continue} interpreter. The trace will be shorter than in Mid-Term 1.
4) 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) \textit{a} and \textit{d}.

2) 
\begin{verbatim}
(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))))
\end{verbatim}

3) 
\begin{verbatim}
interp expr = \{\{\text{fun} \{x\} \{x \ x\}\} \{\text{fun} \{y\} \ 12\}\}
subs = (mtSub)
k = (mtK)

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

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

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

cont val = (closureV \text{'y} \ 12\} = v_2
k = (doAppK v_1 (mtK))

interp expr = \{x \ x\}
ds = (aSub \text{'x} v_2 (mtSub)) = d_1
k = (mtK)

interp expr = x
ds = d_1
k = (appArgk x d_1 (mtK))

cont val = v_2
k = (appArgK x d_1 (mtK))

interp expr = x
ds = d_1
k = (doAppK v_2 (mtK))

cont val = v_2
\end{verbatim}
\[ k = \text{doAppK } v_2 \ (\text{mtK}) \]

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

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

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