Name _____ Section _____

Please do not turn the page until everyone is ready.

Rules:

- The exam is closed-book, closed-notes
- Please stop promptly at 10:20
- There are a total of 60 points, distributed unevenly among the questions
- Please try to write neatly – style matters, but we'll take into account the fact that this is a short exam and it's not always possible to have the time to revise and clean up everything.
- Please keep your answers brief and to the point.

Advice:

- Read questions carefully and understand what's asked before you start writing.
- Leave evidence of thoughts and intermediate steps so you can get partial credit.
- Skip around – if you get hung up on a question, try the next one and come back.
- If you have questions, ask – raise your hand and someone will come to your seat and try to help you out.
- Relax. You are here to learn.

Question 1. (4 points) What are the values of the following Scheme expressions

(a)
```
(define x 1)
(define y 2)
(define z 3)
(let ((x 3)
      (y (+ x 2))
      (z (+ x y 5)))
    (* x z))
```

(b)
```
(define x 1)
(define y 2)
(define z 3)
(let* ((x 3)
       (y (+ x 2))
       (z (+ x y 5)))
    (* x z))
```

Question 2. (4 points) The let construct is not a fundamental Scheme primitive because it can be defined without having it built in to the language. Give a lambda expression that has the same effect as the following let expression:

```
(let ((v1 e1)
      (v2 e2))
   x)
```

Question 3. (3 points) We looked at several parameter passing methods that had different rules for evaluation. Two in particular were call-by-name (thunks) and call-by-need. What is the key difference between these two?

Question 4. (6 points) Recall that in scheme, the function call/cc evaluates a function passing it the current continuation as an argument (i.e., (call/cc (lambda (k) e)). Use call/cc to write a program that loops indefinitely printing the sequence of integers starting from 0 (i.e., 0, 1, 2, 3, …). *Do not* use recursive procedures or assignments.