CS 61A, Spring 2004, Midterm #1

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.

Question 1 (7 points):

What will Scheme print in response to the following expressions? If an expression produces an error message, you may just write "error"; you don't have to provide the exact text of the message. If the value of an expression is a procedure, just write "procedure"; you don't have to show the form in which Scheme prints procedures.

```
(se 'a '(a))
```

```
(word 'a '(a))
```

```
((lambda (x) (bf (bl x))) '(abcd efgh))
```

```
(every (lambda (x) (bf (bl x))) '(abcd efgh))
```

```
(lambda (abcd) (bf (bl abcd)))
```

```
(se (first 'goodbye) (first '(hello)))
```

```
(cond (3 4) (5 6) (else 7))
```

Question 2 (7 points):

Define a procedure `make-closest-value` that takes a nonempty sentence of numbers as its argument. It should return a procedure that takes a single number N as argument, and returns the number from the original sentence that's closest in value to N. (If two numbers from the sentence are equally close to N, you can choose either of them.)

For example:

```
> (define sign (make-closest-value '(-1 1)))
> (sign 20)
1
> (sign -3)
-1

> (define decade (make-closest-value '(0 10 20 30 40 50)))
> (decade 12)
10
> (decade 37)
40
```

You can use the following helper, which takes three numbers as arguments, and returns whichever of `a` and `b` is closer in value to n:

```
(define (closer n a b)
  (if (< (abs (- n a)) (abs (- n b))) a b))
```

Question 3 (7 points):

Write a procedure `values` that takes three arguments: a predicate function of one argument, and two integers. It should return a sentence containing those integers between the two argument integers, inclusive, for which the predicate returns `#t`:

```
> (values even? 20 29)
(20 22 24 26 28)
> (values prime? 10 40)
(11 13 17 19 23 29 31 37)
> (values prime? 24 28)       ; no primes between 24 and 28
()
> (values even? 30 20)        ; 30 > 20
()
```

Don't write helper procedures.

```
(define (values pred from to)
```

_____)

Question 4 (2 points):

What's the order of growth of the procedure `factcount` defined below?

```
(define (factcount sent)
  (factorial (count sent)))                    Θ(__)

(define (factorial n)
  (if (= n 0)
      1
      (* n (factorial (- n 1)))))
```

Question 5 (4 points):

Consider the following procedure definition and invocation:

```
(define (f a b c)
  (if (= a 3) b c))

(f (+ 1 2) (+ 3 4) (+ 5 6))
```

(a) The expression `(+ 1 2)` will be evaluated under:

normal order

applicative order

both normal and applicative order

neither normal nor applicative order

(b) The expression `(+ 5 6)` will be evaluated under:

normal order

applicative order

both normal and applicative order

neither normal nor applicative order

Question 6 (6 points):

(a) Does the following procedure generate an iterative or recursive process?

```
(define (seq n)
  (define (help n k sent)
    (cond ((= n 0) sent)
          ((> k n) (help (- n 1) 1 sent))
          (else (help n (+ k 1) (se sent k)))))
  (help n 1 '()))
```

(b) What value is returned by `(seq 4)`?

(c) If your answer to part (a) is "iterative," rewrite the procedure to generate a recursive process. If you answer is "recursive," rewrite the procedure to generate an iterative process.

Question 7 (6 points):

Suppose you've written a program `Arabic` that is supposed to convert Roman numerals to Arabic numerals, like this:

```
> (Arabic 'MMIV)
2004
```

[In case you're unfamiliar with Roman numerals: Each letter has a numeric value, I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000. The values for all the letters are added, except that if a smaller one comes before a bigger one, the smaller one is subtracted from the larger. In the example above, MMIV is 1000 + 1000 + (5 – 1).]

Your program's actual behavior is as follows:

```
> (arabic 'MMIV)
2004

> (arabic 'MLXVI)
1066
```

```
> (arabic 'D)
500

> (arabic 'MCDXCII)
2092
```

(a) What kind of arguments will produce incorrect results? Be as specific as possible. ("Specific" doesn't mean to use a lot of words; one sentence should be enough. An example would be, "Arguments containing the letter M fail," although that's not the actual answer.)

(b) What incorrect result is returned for such arguments? Again, be as specific as possible.

(c) What is the most likely error in the definition of the procedure?