Midterm Sample Solutions

Sample solutions to selected questions.

1. (20 marks)

1.1 [7 marks] Consider the following Lisp definition.

```
(defun s (L x v)
    (cond
       ((null L) nil)
       ((atom L) (if (eq L x) v L))
       (t (cons (s (car L) x v) (s (cdr L) x v)))
    )
)
```

For each function call below, show the result.

(a) `(s '(x q x x) 'x 'w)`

Answer: `(w q w w)`

(b) `(s '(x q x x) 'x '(1 2))`

Answer: `((1 2) q (1 2) (1 2))`

(c) `(s '((a b x) (x) ((q (x) x))) 'x 'w)`

Answer: `((a b w) (w) ((q (w) w)))`

1.2 [4 marks] What will be returned when the following expression is evaluated

```
(mapcar
       '(lambda (x) (if (> x 1) (- x 1) (+ x 1)))
       (reduce 'append '((1 2) (3 4) (5 6)))
)
```

where the append function is defined as usual.

Answer: `(2 1 2 3 4 5)`

1.3 [9 marks] Consider the following Lisp definitions.

```
(defun f (L g)
     (if (null L)
         nil
         (if (null (cdr L))
             (cons (funcall g (car L) (car L)) nil)
             (cons (funcall g (car L) (cadr L))
                   (f (cddr L) g))
         )
      )
)

(defun g (X Y)
     (cons X (cons Y nil))
)


(defun h (X Y)
     (cons X Y)
)
```

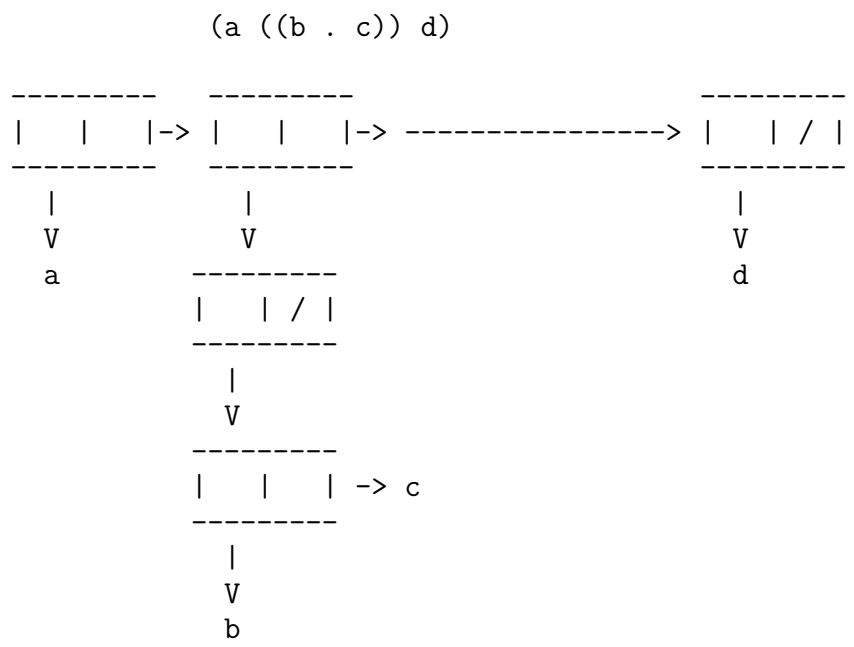
3 Recall that `(funcall g arg1 ... argn)` is to apply function `g` to the arguments that follow.

For each call below, show what will be returned.

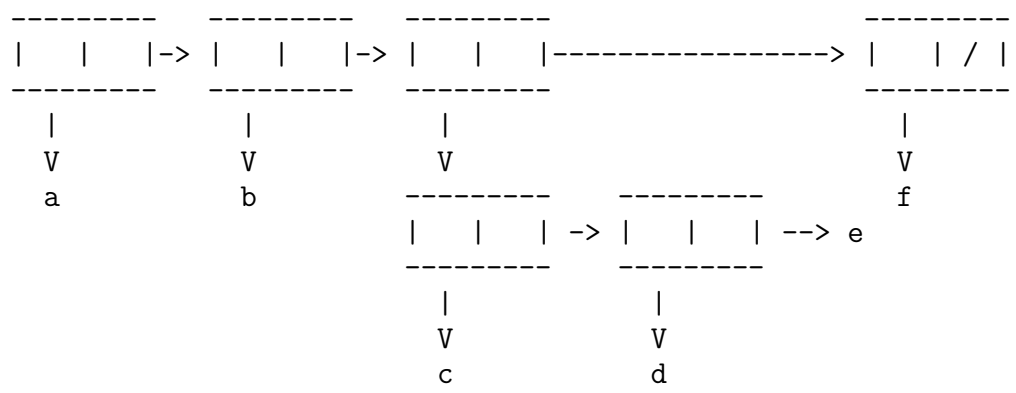
```
(i)      (f '(a b c) 'g)

Answer: ((a b) (c c))

(ii)     (f '(1 2 3 4 5 6) 'g)

Answer: ((1 2) (3 4) (5 6))

(iii)    (f '(1 2 3 4 5 6) 'h)

Answer: ((1 . 2) (3 . 4) (5 . 6))
```

2. (8 marks) Machine level representation.

Show the machine level representation of the following s-expression.



Show the simplest s-expression that is stored internally by the following structure:



answer: (a b (c d . e) f)

3. (20 marks)

3.3 [7 marks] Define a function

```
(defun sum (L) ...)
```

which takes a (possibly nested) list of integers, and returns the sum of all the integers in it. E.g.,

```
(sum '(3 (4) ((5) 1))) ==> 13


(defun sum (L)
    (if (null L)
        0
        (if (atom L)
            L
            (+ (sum (car L)) (sum (cdr L)))
        )
    )
)
```

4. (17 marks)

4.1 [5 marks] Recall that T (true) and F (false) in lambda calculus can be encoded as follows:

$T = (\lambda xy \mid x)$
$F = (\lambda xy \mid y)$

Reduce the following lambda expression to its normal form. Show all the steps.

$$\begin{aligned} & F\ T\ F\ F\ T\ T\ T \\ \rightarrow\ & F\ F\ T\ T\ T \\ \rightarrow\ & T\ T\ T \\ \rightarrow\ & T \end{aligned}$$

4.2 [6 marks] Reduce the following λ-expression to its normal form. You may use any reduction strategy. Show all the steps.

$$\begin{aligned} & (\lambda xy \mid x(yyy))\ (\lambda x \mid xy)\ (\lambda x \mid x) \\ \rightarrow\ & (\lambda x \mid xy)((\lambda x \mid x)(\lambda x \mid x)(\lambda x \mid x)) \\ \rightarrow\ & (\lambda x \mid xy)((\lambda x \mid x)(\lambda x \mid x)) \\ \rightarrow\ & (\lambda x \mid xy)(\lambda x \mid x) \\ \rightarrow\ & (\lambda x \mid x)\ y \\ \rightarrow\ & y \end{aligned}$$

4.3 [6 marks] Someone argues that one can make recursive calls in lambda calculus without using the Y combinator, at least for the SUM example, for which all we need to do is to define it to be

$$SUM = (\lambda\ f\ n \mid (ZEROT\ n)\ 0\ (+\ n\ (f\ f\ (-\ n\ 1))))$$

Recall that ZEROT is the lambda expression for testing whether its argument is zero or not, and +, -, and integers such as 0, 1, 2, etc. denote the corresponding lambda expressions that encode addition, subtraction, and natural numbers, respectively. The details of these encodings are unimportant in this question.

In this question, you are asked to verify whether this idea works or not by showing the details of computing $3 + 2 + 1$ from the expression $SUM\ SUM\ 3$.

$SUM\ SUM\ 3$
$\rightarrow (+\ 3\ (SUM\ SUM\ 2))$
$\rightarrow (+\ 3\ (+\ 2\ (SUM\ SUM\ 1)))$
$\rightarrow (+\ 3\ (+\ 2\ (+\ 1\ (SUM\ SUM\ 0))))$
$\rightarrow (+\ 3\ (+\ 2\ (+\ 1\ 0)))$
$\rightarrow 6$

5. (8 marks) In this problem, when a question is about context-based interpretation, we use the notation $\{x_1 \rightarrow v_1, ..., x_m \rightarrow v_m\}$ for context, and $[Fn, CT]$ for closure where Fn is a lambda function and CT is a context.

Consider the problem of evaluating the following lambda expression in some context denoted CT0.

```
((lambda (f y) (f y y)) (lambda (x z) (+ x (+ z 2))) 3)
```

(a) Show the context when the subexpression `(f y y)` is being evaluated.

Answer: `{f ->[(lambda (x z) (+ x (+ z 2))), CT0], y -> 3} U CT0`

(b) Show the context when the subexpression `(+ x (+ z 2))` is evaluated.

Answer: `{x -> 3, z -> 3} U CT0`

6. (7 marks) Recall the definition of the following stack operations

```
SEL  (x.s) e (SEL ct cf.c) d   ->  s e c' (c.d)
                 where c' = ct if x is T, and c' = cf if x is F
JOIN  s e (JOIN.c) (cr.d)  ->  s e cr d
```

Show all the state changes on the SECD machine for the execution of the code in the control stack. Note that > below is the familiar primitive greaterThan function.

```
    s   e   (LDC 5 LDC 2 > SEL (LDC 9 LDC 5 + JOIN) (LDC 7 JOIN) . c)  d
-> (5.s) e (LDC 2 > SEL (LDC 9 LDC 5 + JOIN) (LDC 7 JOIN) . c)  d
-> (2 5.s) e (> SEL (LDC 9 LDC 5 + JOIN) (LDC 7 JOIN) . c)  d
-> (F.s) e (SEL (LDC 9 LDC 5 + JOIN) (LDC 7 JOIN) . c)  d
-> s e (LDC 7 JOIN)  (c.d)
-> (7.s) e (JOIN) (c.d)
-> (7.s) e c d
```