

## CS 401/501 EXAM #2 SAMPLE QUESTIONS (SOLUTION)

1. The mathematical model of computation that denotational semantics is based upon is called lambda calculus.
2. True or False. Denotational semantics is a method for defining the semantics of a programming language by giving an interpreter for that language.

False. Denotational semantics defines a mapping from syntax domains to functions over semantic domains. These functions may be executed to effect an interpreter.

3. Consider the following Lisp function:

```
(defun update (store V n)
  (prog (currentenv newbinding newstore)
    (setq currentenv (cdaddr store))
    (setq newbinding (cons (list 'eq 'V (list 'quote V)) (list n)))
    (setq newstore (list 'lambda (list 'V) (cons 'cond (cons newbinding currentenv))))
    (return newstore)
  )
)
```

Note that the (prog (list of local variables) (exp-1) (exp-2) ... (exp-n) (return exp)) construction declares a list of local variables used in the expressions with exp being the value returned. Let S represent the Lisp expression (lambda (V) (cond ((eq V (quote x)) 4) (t (quote bottom)))). What is the internal list representation that Lisp uses for S? What is (cdaddr S)?

```
((EQ V 'X) 4) (T 'BOTTOM))
```

Trace the execution of (update S 'x 5), showing the values of currentenv, newbinding, and newstore.

```
>(setq currentenv (cdaddr store))
((EQ V 'X) 4) (T 'BOTTOM))
>(setq newbinding (cons (list 'eq 'V (list 'quote 'x)) (list 5)))
((EQ V 'X) 5)
>(setq newstore (list 'lambda (list 'V) (cons 'cond (cons newbinding currentenv))))
(LAMBDA (V) (COND ((EQ V 'X) 5) ((EQ V 'X) 4) (T 'BOTTOM)))
```

4. Consider the Prolog program which solves the Towers of Hanoi problem. This problem is to move a stack of blocks from one location to another without ever changing the order in which blocks are placed on top of each other. A third stack may be used but only one block may be moved at a time.

```
hanoi(N) :- move(N, left, center, right).
```

```
move(0, X, Y, Z).
```

```
move(N, X, Y, Z) :-
```

```
    N > 0, M is N - 1, move(M, X, Z, Y), display([X,Y]), move(M, Z, Y, X).
```

Trace the above program using the query `hanoi(3)`. Show all queries which are generated by the execution and all moves which are printed. You may wish to draw a tree to show this clearly.

```
Script started on Sat 11 Mar 2006 10:07:30 AM CST
```

```
vulcan6% xsb
```

```
[xsb_configuration loaded]
```

```
[sysinitrc loaded]
```

```
XSB Version 2.7.1 (Kinryo) of March 5, 2005
```

```
[i686-pc-linux-gnu; mode: optimal; engine: slg-wam; gc: indirection; scheduling: local]
```

```
| ?- [hanoi].
```

```
[Compiling ./hanoi]
```

```
++Warning[XSB]: [Compiler] ./hanoi: Singleton variable X in a clause of move/4
```

```
++Warning[XSB]: [Compiler] ./hanoi: Singleton variable Y in a clause of move/4
```

```
++Warning[XSB]: [Compiler] ./hanoi: Singleton variable Z in a clause of move/4
```

```
[hanoi compiled, cpu time used: 0.0140 seconds]
```

```
[hanoi loaded]
```

```
yes
```

```
| ?- trace.
```

```
yes
```

```
[trace]
```

```
| ?- hanoi(3).
```

```
    (0) Call: hanoi(3) ?
```

```
    (1) Call: move(3,left,center,right) ?
```

```
    (2) Call: move(2,left,right,center) ?
```

```
    (3) Call: move(1,left,center,right) ?
```

```
    (4) Call: move(0,left,right,center) ?
```

```
    (4) Exit: move(0,left,right,center) ?
```

```
    (5) Call: display([left,center]) ?
```

```
[left,center]    (5) Exit: display([left,center]) ?
```

```
    (6) Call: move(0,right,center,left) ?
```

```
    (6) Exit: move(0,right,center,left) ?
```

```

(3) Exit: move(1,left,center,right) ?
(7) Call: display([left,right]) ?
[left,right] (7) Exit: display([left,right]) ?
(8) Call: move(1,center,right,left) ?
(9) Call: move(0,center,left,right) ?
(9) Exit: move(0,center,left,right) ?
(10) Call: display([center,right]) ?
[center,right] (10) Exit: display([center,right]) ?
(11) Call: move(0,left,right,center) ?
(11) Exit: move(0,left,right,center) ?
(8) Exit: move(1,center,right,left) ?
(2) Exit: move(2,left,right,center) ?
(12) Call: display([left,center]) ?
[left,center] (12) Exit: display([left,center]) ?
(13) Call: move(2,right,center,left) ?
(14) Call: move(1,right,left,center) ?
(15) Call: move(0,right,center,left) ?
(15) Exit: move(0,right,center,left) ?
(16) Call: display([right,left]) ?
[right,left] (16) Exit: display([right,left]) ?
(17) Call: move(0,center,left,right) ?
(17) Exit: move(0,center,left,right) ?
(14) Exit: move(1,right,left,center) ?
(18) Call: display([right,center]) ?
[right,center] (18) Exit: display([right,center]) ?
(19) Call: move(1,left,center,right) ?
(20) Call: move(0,left,right,center) ?
(20) Exit: move(0,left,right,center) ?
(21) Call: display([left,center]) ?
[left,center] (21) Exit: display([left,center]) ?
(22) Call: move(0,right,center,left) ?
(22) Exit: move(0,right,center,left) ?
(19) Exit: move(1,left,center,right) ?
(13) Exit: move(2,right,center,left) ?
(1) Exit: move(3,left,center,right) ?
(0) Exit: hanoi(3) ?

```

yes

[trace]

| ?- [ load module push\_io ]

End XSB (cputime 0.03 secs, elapsetime 37.74 secs)

vulcan6% exit

Script done on Sat 11 Mar 2006 10:08:33 AM CST

5. Consider the denotational semantics below:

$$S[\mathbf{S}_1; \mathbf{S}_2] \text{ store} = S[\mathbf{S}_2] (S[\mathbf{S}_1] \text{ store})$$

$$S[\mathbf{V} := \mathbf{E}] \text{ store} = \text{store}[E[\mathbf{E}] \text{ store}/\mathbf{V}]$$

$$S[\mathbf{while} \ \mathbf{C} \ \mathbf{loop} \ \mathbf{S} \ \mathbf{end} \ \mathbf{loop}] \text{ store} = \\ \text{if } C[\mathbf{C}] \text{ store} \text{ then } S[\mathbf{while} \ \mathbf{C} \ \mathbf{loop} \ \mathbf{S} \ \mathbf{end} \ \mathbf{loop}] (S[\mathbf{S}] \text{ store}) \text{ else } \text{store}$$

$$C[\mathbf{E}_1 > \mathbf{E}_2] \text{ store} = \text{if } E[\mathbf{E}_1] \text{ store} > E[\mathbf{E}_2] \text{ store} \text{ then } \text{true} \text{ else } \text{false}$$

$$E[\mathbf{E}_1 + \mathbf{E}_2] \text{ store} = E[\mathbf{E}_1] \text{ store} + E[\mathbf{E}_2] \text{ store}$$

$$E[\mathbf{E}_1 - \mathbf{E}_2] \text{ store} = E[\mathbf{E}_1] \text{ store} - E[\mathbf{E}_2] \text{ store}$$

$$E[\mathbf{I}] \text{ store} = N[\mathbf{I}]$$

$$E[\mathbf{V}] \text{ store} = \text{if } \text{store}[\mathbf{V}] = \perp \text{ then } \top \text{ else } \text{store}[\mathbf{V}]$$

Assume that  $N$  returns the integer value of its argument. Given the initial store  $(\lambda \mathbf{V}.\perp) [1/u] [5/y] [5/z]$ , compute the store which results from evaluating  $S$  on the program below, showing all steps used in the evaluation.

```
while (u > 0) loop
  u := u - 1;
  z := z + y
end loop
```

Let  $\text{store}_0 = (\lambda \mathbf{V}.\perp) [1/u] [5/y] [5/z]$ . The steps of execution using the denotational semantics are:

(a)  $S[\mathbf{while} \ \mathbf{u} > \mathbf{0} \ \mathbf{loop} \ \mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y} \ \mathbf{end} \ \mathbf{loop}] \text{ store}_0$

(b)  $\text{if } C[\mathbf{u} > \mathbf{0}] \text{ store}_0 \text{ then } S[\mathbf{while} \ \mathbf{u} > \mathbf{0} \ \mathbf{loop} \ \mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y} \ \mathbf{end} \ \mathbf{loop}] (S[\mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y}] \text{ store}_0) \text{ else } \text{store}_0$

The following steps elaborate  $C[\mathbf{u} > \mathbf{0}] \text{ store}_0$ .

i.  $\text{if } E[\mathbf{u}] \text{ store}_0 > E[\mathbf{0}] \text{ store}_0 \text{ then } \text{true} \text{ else } \text{false}$

The following steps elaborate  $E[\mathbf{u}] \text{ store}_0$ .

A.  $\text{if } \text{store}_0[\mathbf{u}] = \perp \text{ then } \top \text{ else } \text{store}_0[\mathbf{V}]$

B. 1

The following steps elaborate  $E[\mathbf{0}] \text{ store}_0$ .

A.  $N[\mathbf{0}]$

B. 0

ii.  $\text{true}$

- (c) The following steps elaborate  $S[\mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y}] store_0$ .
- i.  $S[\mathbf{z} := \mathbf{z} + \mathbf{y}] (S[\mathbf{u} := \mathbf{u} - \mathbf{1}] store_0)$
  - ii.  $S[\mathbf{z} := \mathbf{z} + \mathbf{y}] (store_0[E[\mathbf{u} - \mathbf{1}] store_0/\mathbf{u}])$
  - iii.  $S[\mathbf{z} := \mathbf{z} + \mathbf{y}] (store_0[E[\mathbf{u}] store_0 - E[\mathbf{1}] store_0/\mathbf{u}])$   
The evaluation of these 2 terms is identical to that shown in step b-i.
  - iv.  $S[\mathbf{z} := \mathbf{z} + \mathbf{y}] (store_0[0/\mathbf{u}])$   
Let  $store_1 = store_0[0/\mathbf{u}] = (\lambda \mathbf{V}.\perp) [1/\mathbf{u}] [5/\mathbf{y}] [5/\mathbf{z}] [0/\mathbf{u}]$ .
  - v.  $S[\mathbf{z} := \mathbf{z} + \mathbf{y}] store_1$
  - vi.  $store_1[E[\mathbf{z}] store_1 + E[\mathbf{y}] store_1/\mathbf{z}]$   
The evaluation of these 2 terms is identical to that shown in step b-i.
  - vii. Let  $store_2 = store_1[10/\mathbf{z}] = (\lambda \mathbf{V}.\perp) [1/\mathbf{u}] [5/\mathbf{y}] [5/\mathbf{z}] [0/\mathbf{u}] [10/\mathbf{z}]$ .
- (d)  $S[\mathbf{while} \mathbf{u} > \mathbf{0} \mathbf{loop} \mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y} \mathbf{end} \mathbf{loop}] store_2$
- (e) if  $C[\mathbf{u} > \mathbf{0}] store_2$  then  $S[\mathbf{while} \mathbf{u} > \mathbf{0} \mathbf{loop} \mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y} \mathbf{end} \mathbf{loop}] (S[\mathbf{u} := \mathbf{u} - \mathbf{1} ; \mathbf{z} := \mathbf{z} + \mathbf{y}] store_2)$  else  $store_2$
- (f)  $store_2 = (\lambda \mathbf{V}.\perp) [1/\mathbf{u}] [5/\mathbf{y}] [5/\mathbf{z}] [0/\mathbf{u}] [10/\mathbf{z}]$ .