Family name ______________________

Given name(s) ______________________

Student number ______________________

York University

Department of Computer Science & Engineering

CSE 3401 - Functional & Logic Programming

Solutions to Test 2

August 3, 2010

Instructions:
1. The test time is 135 minutes.
2. This is a closed book examination. No examination aids are permitted.
3. If a question is ambiguous or unclear, then please write your assumptions and proceed to answer the question.
4. Return all examinations papers before leaving the exam room.
5. If needed, use extra papers at the end of the booklet. Do not use any other paper.
6. Keep all pages stapled.
7. Keep track of time. Your time is limited.

Good Luck!

Question	Max	Mark
1	15	
2	20	
3	15	
4	10	
5	10	
6	10	
7	10	
8	10	
Total	100	

Question 1. (15 marks)

Assume that **true (T)** is defined as $\lambda xy.x$ and **false (F)** is defined as $\lambda xy.y$. Show that if **AND** is implemented as $\lambda pq.p\ q\ p$, **AND**ing **true** and **false** will results in **false**.

Answer.

$$(\lambda pq.pqp)TF \rightarrow_{\beta}$$
$$((\lambda q.pqp)[p := T])F =$$
$$(\lambda q.TqT)F \rightarrow_{\beta}$$
$$(TqT)[q := F] =$$
$$TFT = (\lambda xy.x)FT \rightarrow_{\beta}$$
$$((\lambda y.x)[x := F])T =$$
$$(\lambda y.F)T \rightarrow_{\beta}$$
$$(F)[y := T] = F$$

Question 2. (20 marks)

Let S be $\lambda x.x\lambda y.yzy$

(a) Restore the dropped parentheses in S (without changing the meaning or structure).

Answer.

$$\lambda x.x\lambda y.yzy = (\lambda x.(x\lambda y.yzy)) = (\lambda x.(x(\lambda y.yzy))) = (\lambda x.(x(\lambda y.((yz)y))))$$

(b) Show the term calculation for it.

Answer.

$$x, y, z, (yz), ((yz)y)$$
$$(\lambda y.((yz)y)), (x(\lambda y.((yz)y)))$$
$$(\lambda x.(x(\lambda y.((yz)y))))$$

(c) Show the set of free variables in S. Show steps for obtaining your answer.

Answer.

$$FV(S) = FV(x(\lambda y.((yz)y))) - \{x\} =$$
$$(\{x\} \cup FV(\lambda y.((yz)y))) - \{x\} =$$
$$(\{x\} \cup (FV((yz)y)) - \{y\}) - \{x\} =$$
$$(\{x\} \cup ((FV(yz) \cup \{y\}) - \{y\}) - \{x\} =$$
$$(\{x\} \cup ((\{y,z\} \cup \{y\}) - \{y\}) - \{x\} =$$
$$\{x,z\} - \{x\} = \{z\}$$

(d) Does S have a β-normal form? If no, why? If yes, what is it?

Answer.

S does not contain any β-redexes and is already in β-normal form.

Question 3. (15 marks)

Assume the following terms have been entered into the LISP interpreter:

```
(setq v1 '((a b c) (d e f)))
(setq v2 '(1 2 3))
(defun f1 (x y z) (+ (* x y) z))
(setq f2 #'(lambda (x y z) (* 100 x)))
(setq f1 v2)
```

How would LISP interpreter answer the following? If there will be an error, mention why.

> (setq w (car (cdr v1)))

(D E F)

> (append (car v1) (cdr v1))

(A B C (D E F))

> (cons v1 ‘v1)

(((A B C) (D E F)) . V1)

> (f1 v2)

Error: F1 expects 3 arguments, not 1.

> (f2 v2)

Error: undefined function F2.

> (apply f1 v2)

Error: The value of F1 is a list, not a function definition.

> (apply ‘f1 v2)

5

> (apply f2 v2)

100

> (funcall f2 v2)

Error: F2 expects 3 arguments, not 1.

> (apply ‘f1 f1)

5

> (mapcar f2 ‘(1) ‘(2) ‘(3))

(100)

> (mapc ‘+ ‘(1 2 3) ‘(4 5 6))

(1 2 3)

Question 4. (10 marks)

Assume the following were entered into top-level of LISP interpreter:

```
(set '|My Library| 20100803)
(setq temp  '((book2) (book1)))
(setf (get '|My Library| 'books)  temp)

(defun addb (book &optional author)
        (setq temp (get '|My Library| 'books))
        (setf (get '|My Library| 'books)
                (cons (list book author) temp)))
```

What would LISP return as a response to each of the following sequence of expressions?

```
> (addb ‘lisp  ‘wilensky)
```

```
((LISP WILENSKY) (BOOK2) (BOOK1))
```

```
> (addb ‘\Prolog )
```

```
((PROLOG NIL) (LISP WILENSKY) (BOOK2) (BOOK1))
```

```
> temp
```

```
((LISP WILENSKY) (BOOK2) (BOOK1))
```

```
> |My Library|
```

```
20100803
```

```
> (symbol-name ‘|My Library|)
```

```
“My Library”
```

Question 5. (10 marks)

(a) Write a function using cond that implements this function:

$$f(x,y) = \left| \begin{array}{ll} 2 & \text{if } y > 0 \text{ and } x - y \geq 0 \\ -2 & \text{if } y < 0 \text{ and } x + y \geq 0 \\ 1 & \text{if } y > 0 \text{ and } x - y < 0 \\ -1 & \text{if } y < 0 \text{ and } x + y < 0 \end{array} \right.$$

Use efficient test expressions (don't check for conditions if not necessary). How many cond clauses do you need?

Answer.

```
(defun f ( x y)
  (cond
        ( (and (> y 0) (>= (- x y) 0))  2)
        ( (and (< y 0) (>= (+ x y) 0)) -2)
        ((> y 0) 1)
        ((< y 0) -1)))
```

Four cond clauses used here. Or it can be done this way:

```
(defun f ( x y)
  (cond
        (  (> y 0) (if  (>=  x y)   2 1))
        (  (< y 0) (if (>= (+ x y) 0) -2 -1))))
```

Two cond clauses used here.

(b) How would you evaluate f(5,6) in LISP and what would you get as a return value? (write the form to ask LISP and the answer given to it)

```
(f 5 6)
```

```
1
```

(c) Since y=0 is not defined in the function, what will be returned by LISP as the value of f(0,0)?

NIL

Question 6. (10 marks)

Use ‘do’ to write an expression that returns a list of the first n odd numbers without creating/effecting any free variables or using append (except n).

For example if n is 3 it returns (1 3 5) and if n is 5, it returns (1 3 5 7 9).

Answer.

I am assuming that n will have a value and the value is greater than or equal to zero:

```
(do  ((i  n (1- i)) (lst nil (cons (1- (* 2 i)) lst)) ) ((zerop  i) lst))
```

Question 7. (10 marks)

(a) Write a recursive function **countnil** to count the number of nil elements in a given list. For example for the list (a b nil 1 x nil 2 nil), it would return 3.

```
(defun countnil (lst)
  (if (null lst) 0
  (if (null (car lst)) (1+ (countnil (cdr lst))) (countnil (cdr lst))) ))
```

Or

```
(defun countnil (lst)
  (cond    ((null lst)  0)
           ((null (car lst)) (1+ (countnil (cdr lst))))
           (t (countnil (cdr lst)))))
```

(b) Write a function **countnilarg** that can accept at least one argument and up to as many as supplied. Then it would call **countnil** in part (a) to count the number of nils in the arguments supplied. Finally it prompts the number of nils in the arguments nicely as follows:

"The number of nil elements is 3"

Note: The input to **countnil** is a list, and the inputs to **countnilarg** are variable number of arguments.

Answer.

```
(defun countnil2 (a &rest b)
        (format nil "The number of nil elements  is ~d" (countnil (cons a b)))
```

Question 8. (10 marks)

We want to write a function readmyfile that reads n symbols from a given file and returns them in a list. We assume that the file has at least n symbols written in it. For example if “data.txt” contains the following:

10 21.5 a (a b) (cons ‘c ‘d) 11 12

We expect

(readmyfile “data.txt” 2) to return

(10 21.5)

The following code is written but it does not work properly.

```
> (defun readMyfile (filename n)
        (let (ins (y nil))
              (setq ins (open filename :direction :input))
              (dotimes  (i n y)
                    (setq y (append  (list (read ins))  y)))
              (close ins)))
```

(a)Using above code, what will be returned for

(readmyfile “data.txt” 2)

And why?

Answer. It will return:

T

Since (close ins) is the last form evaluated and that evaluates to T.

(b) Correct the problems in the code to get

```
> (readmyfile "data.txt"  2)
 (10 21.5)
```

```
(defun readMyfile (filename n)
        (let (ins (y nil))
               (setq ins (open filename :direction :input))
               (dotimes  (i n y)
                       (setq y (append y  (list (read ins)))))
               (close ins)
               y))
```

(c) Given above example file, what do we expect the corrected code to return for

(readmyfile "data.txt" 4)

Answer.
(10 21.5 a (a b))