UNIVERSITI SAINS MALAYSIA

Peperiksaan Semester Kedua
Sidang Akademik 1992/93

April 1993

CSI 502 - Functional Programming

Masa: [3 jam]

ARAHAN KEPADA CALON:

- Sila pastikan bahawa kertas peperiksaan ini mengandungi **LIMA** muka surat yang bercetak sebelum anda memulakan peperiksaan ini.

Answer **ALL** questions.

NOTE : For all function definitions (except for question 1(d)) you are required to use only the following primitive functions : **cons, car, cdr, eq, equal, null, atom, listp,** and any of logical functions such as **and, or,** and **not,** and also any of the arithmetic operations. In addition, you are not allowed to give procedural definitions for any functions, such as using LISP's **prog** function.

1. (a) Determine whether the following pairs of LISP expressions are equivalent in the sense that they evaluate to equivalent S-expressions.

   (i) '(A . (B . (C . NIL))) and '(A (CONS 'B '(C)))

   (ii) (EVAL '(CONS 'A 'B)) and (CAR '((A . B) NIL))

   (iii) '((A . B) X (C . (E F G))) and '((A . B) . (X . ((C . (E F G)))))

   (5/25)

   (b) Give the results of evaluating each of the following expressions.

   (i) (eval '(eval '(car '(cons 'a (b c d)))))

   (ii) ((lambda (x) (apply x '(x (y z)))) '(lambda (r s) (cons r (car s))))

   (3/25)

   (c) For each of the following pair of functions or concepts in LISP, describe briefly the difference (if any) between the given items.

   (i) **set** and **setq**.

   (ii) **apply** and **funcall**.

   (iii) **eq** and **equal.**

   (iv) **free** and **bound** variables.

   (v) **ordinary** and **higher order** functions.

   (10/25)

   (d) Describe how you would represent a group of people with the attributes name, age, and children's names in LISP. Illustrate the use of this representation by defining a few appropriate functions in LISP notations to carry out the following operations on this representation.

   (i) Adding a new person into an existing qroup of people.

   (ii) Finding the total number of children that a particular person has.

   (7/25)

(Note : You can use any LISP dialect, but make sure that you specifically mention which in your answer)

2. (a) Define the functions which are described in the following either in LISP or a functional design language:

(i) **intersect**(X,Y,Z) which takes an atom Z and linear lists X and Y (which may be nested) as arguments and returns a list which consists of the common elements of X and Y, not including any occurrence of atom Z. The elements in the resulting list must appear in the order they appear in X followed by Y and should not consist any duplication of elements.

Example :

(**intersect** '(a b a d e) '(f a d b) 'd) returns (a b)

(ii) **rotate_right**(X,N) which takes a linear list X and an integer N as arguments and returns the list X with its has been rotated N positions to the right whenever N is equal or less than the number of the elements of X. Otherwise, return X as result.

Examples :

(**rotate_right** '(a b a d e) 3) returns (e d a a b)
(**rotate_right** '(a b c) 3) returns (c b a)
(**rotate_right** '(a b) 3) returns (a b)

(iii) **substitute**(X,Y, Z) which takes two linear lists X and Y and an S-expression Z as arguments and return the list X with each occurrence of its (first-level) elements that are identical to list Y have been replaced by Z. However, if X is nil then return the atom **empty**.

Examples :

(**substitute** '(a (b c (d e)) X (d e)) '(d e) '(d . e))
returns (a (b c (d e)) X (d . e))

(24/40)

(b) Without using any free variables, use the accumulating parameter technique to define the following functions. Show also how to properly invoke these functions.

(i) **remove**(X,A) which takes a linear list X and an atom A and remove all but the first occurrence of A in X, if such number of occurrences is more than one. Otherwise, if there is exactly one occurrence of A, retain that one.

Examples :

(**remove** '(a b b c d e d f g) 'd) returns (a b b c e d f g).
(**remove** '(a b d) 'd) returns (a b d).

(ii) **truth_value**(X) which takes a linear list that represents a conjunction of propositional variables. It will return the truth value of such conjunctive expression. You can assume that there exists a function **val** where val(Y) returns the truth value of atom Y.

Examples :

(**truth_value** '(P Q R T)) returns T when all (val 'P) and (val 'Q) and (val 'R) and (val 'T) return the value T.

(16/40)

3. (a) Modify the following LISP expression so that **setq** will be eliminated through the use of lambda expression.

```
(defun price (p)
  (cond
    (( < (setq temp (length p)) 10) 'cheap)
    ((< temp 100)  '(not bad))
    ( t  'expensive))))
```

(2/35)

(b) Describe what happens when the expression (**tricky** nil) is evaluated where the function **tricky** is defined as follows :

```
(defun tricky (x)
  (cond ((not x) ((lambda (y) (tricky y)) (read)))
        (t 'again)))
```

(5/35)

(c) Consider the following higher-order function :

map (f) = λ (x) if null(x) then NIL else
cons(f(car(x)), (map(f))(cdr(x)))

(i) Verify through type-checking that map and map(map(f)) can be of type (X -> Y) - > (List(X) -> List(Y)) and List(List(X)) -> List(List(Y)) respectively.

(ii) What is the result of evaluating (map (λ (y) y + y)) (x) when x is bound to the list (3 6 8) ? Show how you obtain this answer.

(iii) What would be the type of map(f) if f is of type X -> List(X) ? Justify your answer. Suppose that g = λ (z) cons(z,z). What would be the result of evaluating (map (g)) (x) when x is bound to the list ((a b) (d e)) ? Show how you obtain this answer.

(20/35)

(d) Simplify the following λ-calculus λ-expressions to the simplest form possible. For each case state the type of conversion rules that you use at each transformation step.

(i) $(\lambda x . ((\lambda f . (\lambda x . (f\ 3))\ 5)\ (\lambda y . x))\ 12$

(ii) $(\lambda f g x . f x (g x))\ (\lambda x y . x)\ (\lambda x y . x)$

(3/35)

(e) In λ-calculus, what is meant by a valid substitution E [E'/V] ? Show by giving an example each, which shows that if substitutions in α-conversion and β-conversion rule are allowed to be invalid, then it follows that any two λ-expressions are equal.

(5/35)

- oooOooo -