

BILKENT UNIVERSITY

DEPARTMENT OF COMPUTER ENGINEERING

CS315 PROGRAMMING LANGUAGES

HOMEWORK 3

BERAT BİÇER

21503050

23.12.2017

ANKARA

**Question 1**

Source:

; Callee Function, fcall

(define (fcall list1 list2 list3)

(if (null? list1)

list3

(fcall (cdr list1) (cdr list2) (append list3 (list (cons (car list1) (car list2)))))))

; Caller Function, pair-up

(define (pair-up list1 list2)

(fcall list1 list2 '()))

This program uses tail recursion to form an association list from the function parameters. It is assumed that the input is valid, and therefore no error-checking is done.

Caller function accepts 2 arguments: 2 lists with the same size. It is also assumed that the input satisfies this condition. Then, the callee function is invoked with these parameters and an empty list.

In the callee, the base case checks whether the first list is empty. If so, this means the input is exhausted and the 3rd argument can be returned. If not, a recursive call is made to the callee using cdr of the 1st and 2nd lists, and the list acquired by appending the pair of car of 1st list and car of 2nd list to the 3rd list.

For the sample call given with the assignment, (pair-up ‘(a b c) (list 1 2 3)), output is as follows:

((a . 1) (b . 2) (c . 3))

The output from my local machine is given at the end of this file, together with the second question.

**Question 2**

Source:

(define (deep-remove item ls)

(cond

((null? item) #f)

((list? item) #f)

((null? ls) '())

((and (not (list? (car ls))) (equal? (car ls) item)) (deep-remove item (cdr ls)))

((and (not (list? (car ls))) (not (equal? (car ls) item))) (cons (car ls) (deep-remove item (cdr ls))))

((list? (car ls)) (cons (deep-remove item (car ls)) (deep-remove item (cdr ls))))))

This function takes 2 arguments: an atom and a list, and returns the list containing the exact structure of the argument list except the given atom is removed completely.

Function first checks whether the given atom is in fact and atom. If not, it returns false. Then, the base case, which is the argument list is null, is checked. If not, the function checks whether the car of the argument list is an atom. If it is and it is also equal to the argument atom, without any concatenation, the function makes a recursive call to itself with the cdr of the argument list and the argument atom. This way, we do not include the car of the argument list to the returned structure if it equal to the given atom. Also, we use the function equal, not eq or eqv, to compare the structural and contextual equality of its arguments.

If the car of the argument list is an atom and it is not equal to the argument atom, we concat the car of the argument list with the result of the recursive call made to fcall using the argument atom and cdr of the argument list. This ensures we concatenate the car of the argument list to the rest of the structure, or on other words, we preserve the structure of the argument list.

Lastly, we check if the car of the argument list is a list. If so, we concatenate the results of the recursive calls made to fcall using argument atom along with car of the argument list and cdr of the argument list. This is equivalent to invoking the function to the head of the list and to the rest of the list, as if we are in an iterative language. By concatenating the results, as before, we preserve the structure of the list.

The output to the sample in the assignment, (deep-remove ‘a ‘(x a (b a (g a g) c (a)))), is as follows:

(x (b (g g) c ()))

The outputs from my computer are as follows, as a screenshot of Ubuntu console:

