CMPUT 325 LEC B1 - Winter 2021 - NON-PROCEDURAL PROG LANGUAGES

Assignment 1 (due Feb 2nd)

This assignment is due Tuesday Feb. 2nd at 11:55pm. **NO LATE SUBMISSIONS.** This assignment should be submitted as a single text file. The filename should be (your student ID#).lisp, for example, 1234567.lisp

Before submitting, **make sure your program runs correctly on our lab machines**.

Each question should be preceded with a comment line clearly indicating that the code for that question is starting. Minor functions can be reused in later questions. You must follow the programming style and marking guideline when documenting your work.

The beginning of the file you hand in should look like this:

```
;QUESTION 1
;documentation
(defun xmember (X Y)
...
)
;documentation
(defun minor_helper_function_for_question_1 (...)
...
)
...
;QUESTION 2
;(question 1 should not use any functions beyond this point)
...
etc
```

Assignment Marks:
This assignment is worth 10 marks. Your programs must be readable and understandable as well as correct. You should read the guidelines as given in programming style and marking guideline. You can lose up to half of the marks for bad style even if your programs are correct.

Restrictions:

Only the following built-in Lisp functions and special forms may be used:

```
(atom x)
(null x)
(eq x y)
(equal x y)
(numberp x)
(append x y)
(car x)
(cdr x)
(cons x y)
(if x y z)
(cond ... )
(let ((x y) (u v)) z)
(let* ((x y) (u v)) z)
(defun ...)
(quote x) and its short form 'x
(list x1 x2 ...)
(print ...)
(sort L fun) % this is useful for the last problem
```

and numeric operators and comparisons, and logic connectives such as

```
(+ x y)
(- x y)
(* x y)
(/ x y)
(< x y)
(> x y)
(= x y)
(<= x y)
(>= x y)
(and x y)
(or x y)
(not x)
```

You may also use a combination of car and cdr, such as

```
(cadr ...), (cdaar ...)
```

etc.

You may write one or more functions to solve any given problem below. In some cases, it is desirable to decompose a problem into some smaller ones. However, if a problem has a straightforward solution, it's a bad idea to solve it in a complex way by decomposition.

#1 (1 mark)

Write the Lisp function:

```
(xmember X Y)  Examples
```

It returns T if argument X is a member of the argument list Y and NIL otherwise. This should also test for lists being members of lists. Both the argument X and the list Y may be NIL or lists containing NIL. See the examples. You cannot just call the built-in function member (see restrictions above).

#2 (1 mark)

Write the Lisp function:

```
(flatten x)  Examples
```

where the argument x is a list with sublists nested to any depth, such that the result of (flatten x) is just a list of atoms with the property that all the atoms appearing in x also appear in (flatten x) and in the same order. In this question, you may assume that NIL and () will not appear in the given list x

#3 (1 mark)

Write the Lisp function:

```
(remove-duplicate x)  Examples
```

It takes x as a list of atoms and removes repeated ones in x. The order of the elements in the resulting list should preserve the order in the given list.

#4 (1 mark)

Write the Lisp function:

```
(mix L1 L2)  Examples
```

that mixes the elements of L1 and L2 into a single list, by choosing elements from L1 and L2 alternatingly. If one list is shorter than the other, then append all remaining elements from the longer list at the end.

#5 (2 marks)

Write the Lisp function:

```
(allsubsets L)  Examples
```

that returns a list of all subsets of L. How the subsets in the resulting list are ordered is unimportant.

Hint: Use an accumulator to accumulate all subsets of L, e.g., as

```
(defun allsubsets (L)
      (gen-subsets (cons nil nil) L))
```

where the first parameter for gen-subsets is an accumulator that starts with an empty set, which is a subset of any set.

#6 (4 marks)

A web page A containing a link to another one B is represented by a pair, (A B). Give a list L of such pairs, write two Lisp functions:

```
(reached x L)
```

where x is a web page, L is a list of pairs representing linkage, and the function returns a list of all web pages that can be reached from x (x should not be part of the result). The order of the web pages in the resulting list is unimportant.

The importance of a web page could be determined by how many other web pages refer to it. A web page A is said to refer to another web page B iff A contains a (direct) link to B, and A and B are not the same web page (i.e., a web page referring to itself doesn't count). Multiple links from (A,B) count as one for the importance of web page B.
Define a function

```
(rank S L)
```

where S is a list of atoms naming web pages, and L is a list of pairs representing linkage. The function returns a permutation of S such that the web pages are ordered according to the criterion above, i.e., the most referred web page is the first in the list, and so on. If two web pages are equally important in terms of references, then it doesn't matter how they are ordered.

Hint: Count the number of references to each atom in S to get a list, say

```
((Cmput325 23) (UofA 128) (CSD 68))
```

Then, you can tailor the built-in function sort for your own needs, for example, by
defining

```
(defun mySort (L)
    (sort L 'greaterThan))

(defun greaterThan (L1 L2)
    (> (cadr L1) (cadr L2)))
```

This will give you, for the above example,

```
((UofA 128) (CSD 68) (Cmput325 23))
```

from which you can get the final result

(UofA CSD Cmput325)

Note**: sort is destructive** - it changes the argument list given. So **sort is NOT pure functional** in Lisp. If you want to save the input list, you can easily define a copy function that gives you a fresh copy of the input list.

Submission status

Attempt number	This is attempt 1 (1 attempts allowed).
Submission status	No attempt
Grading status	Not graded
Due date	Tuesday, 2 February 2021, 11:55 PM
Time remaining	7 days 6 hours
Last modified	-
Submission comments	Comments (0)

Add submission

You have not made a submission yet.