CMPUT 325 Wi17 - NON-PROCEDURAL PROG LANGUAGES Combined LAB LEC Wi17

Assignment 1

This assignment is due Thursday Feb 2 at 23:55pm. **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**.

Link to submission page

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
```

`(copy-list L) % useful in conjunction with sort, see below.` This function creates a fresh copy of any input lists. For documentation see http://clhs.lisp.se/Body/f_cp_lis.htm#copy-list

Note about sort**: sort is destructive** - it changes the argument list given. So **sort is NOT pure functional** in Lisp. If this causes problems in your code:

- Either avoid using sort

- Or use sort only on a copy of the list, as in

```
(sort (copy-list L) sort-predicate)
```

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 Y is a member of the argument list X and NIL otherwise. This should also test for lists being members of lists. Both the list X and the argument 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:

```
(mix L2 L1)  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 elements from the longer list at the end.

#4 (2 marks)

Write the Lisp function:

```
(split L)  Examples
```

that splits the elements of L into a list of two sublists (L1 L2), by putting elements from L into L1 and L2 alternatingly.

#5 (1 mark)

Answer the following two questions in English (no programming). Put your answers into a comment in your submission file.

#5.1
Let L1 and L2 be lists. Is it always true that (split (mix L2 L1)) returns the list (L1 L2)? If yes, give a proof. If no, describe exactly for which pairs of lists L1, L2 the result is different from (L1 L2).

#5.2
Let L be a list. Is it always true that (mix (cadr (split L)) (car (split L))) returns L? If yes, give a proof. If no, describe exactly for which lists L the result is different from L.

#6 (4 marks plus 1 possible bonus mark)

Write a Lisp function to solve the **subset sum** problem: given a list of numbers L and a sum S, find a subset of the numbers in L that sums up to S. Each number in L can only be used once.

(subsetsum S L) Examples Harder examples

The result should be a list (n1 n2 ... nk) such that n1 + n2 + ... nk = S and all numbers n1,...,nk are in L. The numbers n1,...,nk should be in the same order as they appear in L. The result should be nil if the problem has no solution. The problem may have more than one solution. Any valid solution is OK.
Hint: the basic recursive solution strategy is to try two cases for each number in L: 1. try to include the number in the sum, and 2. try to not include the number in the sum.
Warning: for difficult instances of L and S, this function can take a long time to execute. A good solution will eliminate many hopeless cases in the search. We will have a variety of test cases, starting from very simple to hard. We will use a timeout of 30 seconds for each test case.

A reasonably efficient, correct implementation with good documentation will get the full 4 marks. For the most efficient programs, up to 1 bonus mark will be awarded. These will be judged by using relatively large and difficult test instances. We may use a longer timeout to decide about the bonus mark.

1. The input list L contains only positive integers. So L will not contain entries such as -4 or 0 or 1.345. The sum S will also be a positive integer.

2. **Do not use the built-in function eq for numbers. Use = instead**. See for example this blog post on Equality.

P.S.: Do not try posting homework on the internet! Do your own work, and ask TA for help if needed.

End of Assignment 1.

Last modified: Saturday, 21 January 2017, 12:04 AM