CSCE 476/876 Spring 2004 Recitation exercises

Yaling Zheng
Constraint Systems Laboratory
University of Nebraska-Lincoln
yzheng@cse.unl.edu

February 11, 2004

- Exercises from 1 to 9 are excerpted from the Online Lisp Training of by Franz Inc.
- Exercise 14 are excerpted from Artificial Intelligence A Modern Approach (SECOND EDITION) of Stuart Russell and Peter Norvig, page 124.

---

1 "Hello World"

Create a source file, write a function that prints "Hello World". Save the file. Compile the file. Load the `.fasl` file. Run the function.

2 Arithmetic

Write a function named polynomial.

- It must take 4 arguments: a, b, c and x.
- It must return the value of $ax^2 + bx + c$.

3 My-compile-and-load

Write a function that compiles and loads a file of Common Lisp source code.

- It takes as argument a file name-string.
- Returns the symbol t.

4 Functions first, second, ..., nth

1. Define a function that
   - takes as argument a list
   - returns a list of the second and fourth elements of its argument.

   For example, for a list (9 7 6 5), this function returns (7 5).
   Note: Use two methods,

2. Define a function that
   - takes as arguments 2 non-negative integer and a list
   - displays the number of elements in the list argument (use the length function), and
   - returns a list of the elements of the list argument at the indices specified by the other 2 arguments.

   For example, for integer 2 3 and a list (1 4 5 7), this function
   - displays "There are 4 elements in (1 4 5 7)"
   - and returns: (5 7)

5 Function rest

Define a function that

- takes as arguments a list and
- returns a list of the result of calling first on the list argument and the result of calling rest on the list argument.

For example, for a list (2 3 4 5) this function returns (2 (3 4 5))

6 Function member

Define a function that takes as argument an object and a list, and returns that part of the list which follows the matching object. For example,

- for an object is 3, a list is (2 3 4 5 6 7)
- it returns (4 5 6 7)

7 Optional arguments to functions

Define a function that

- takes 2 required arguments and 2 optional arguments, and
- Prints each of its arguments.

Test your function as follows:

- Call the function with 2 non-integer arguments at least once.
- Call the function with 3 arguments, not all of them integers, at least once.
- Call the function with 4 arguments at least once.

8 Keyword arguments to functions

Example of keyword arguments:

```
(defun foo (&key (a 0) b c d (e 0) (f 0) (g 0) h i (j 0))
  ;; keyword args default to nil, unless another default is
  ;; specified, as with a, e, f, g, and j.
 ...)
```

In the call `(setf x (foo :e 5 :j 32))` the unspecified parameters are set to their default values. Note that the parameters can be specified in any order. Define a function `doit` that

- takes 2 required arguments alpha and beta and 2 keyword arguments gamma and delta, and

- prints each of its arguments.

To test your function with the following calls:

```
(doit 2 3)
(doit 2 3 :alpha 5)
(doit 2 3 :beta 7 :alpha 5)
```

9 Rest arguments to functions

Define a function doit2 that

- takes 2 required arguments alpha and beta and an &rest argument all-the-others, and
- prints each of its arguments.

Test your function with the following calls

```
(doit2 2 3)
(doit2 2 3 5)
(doit2 2 3 'July 4 1776)
(doit2 3 4 5 6 7)
```

10 Show Lisp package

There is a Common Lisp function `list-all-packages`. Use Composer or the on-line documentation of Common Lisp to learn what this function does. Then

- Define a function that prints the name of every package for which the first element of the list of packages it uses is the Common Lisp package.
- Define a function that prints the name of every package for which one of the packages it uses is the Common Lisp package.

11 Search an element in a list

Define a function that searches for an element in a list. When the test succeeds the function should return that part of the list of which the element found is the first element.

12 Find a keyword in a list

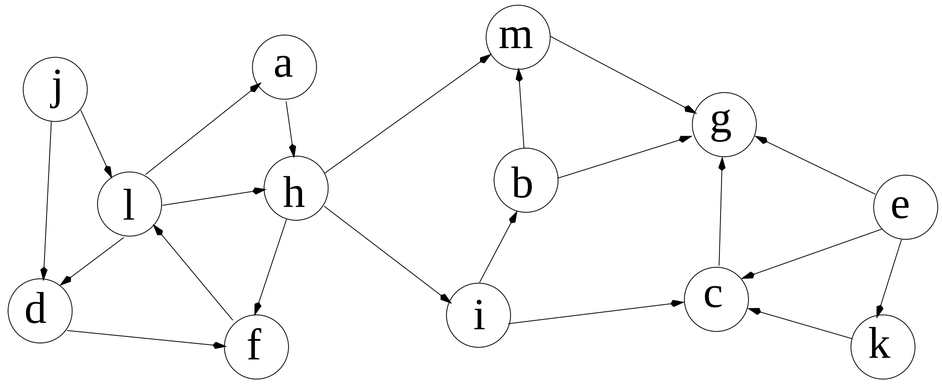
Define a function that takes as arguments a symbol in the keyword package, a list of which alternating elements (specifically the first, third, fifth, ...) are keywords. The function should attempt to match the first argument with the first, third, fifth, ... elements of the list. When the test succeeds, the function should return the list elements after the one for which the match was made.

13 Find a object in a list of lists

Define a function that takes as arguments any lisp object and a list of lists. The function should match the first argument with the first elements of each of the sublists in the second argument. As soon as a match is made, the function should return the sublist for which the match succeeded.

14 Breadth-first traverse in a directed graph

Given a directed graph, commit breadth-first search from a vertex.



Hint: define a structure to represent each node and its neighbors.

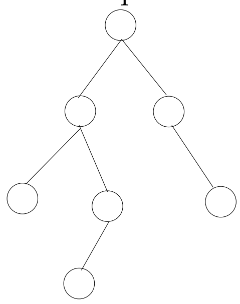
```
(defstruct graph-node
    name
    neighbor
    visited-flag )
```

15 Find a path from one vertex to another in a graph

Given a directed graph and two vertices, u, v, write a function that whether v can be reached from u. When this is the case, return the path from u to v.

16 Depth-first traversal of a tree

Given a tree and its root, write a function that traverses the tree from the root in a depth-first manner.



Hint: define the structure of a tree, e.g.,

```
(defstruct tree-node
    name
    parent
    children
    visited-flag)
```

17 Determine whether a graph is bipartite

Given a graph, write a function that determines whether or not it is a bipartite graph.

18 A Simple Maze Problem, AIMA pp. 124

Consider the maze in page 124 in AIMA (2nd edition). Assuming that the agent knows nothing of the environment, implement an agent that takes

- a maze description, including boundaries

- an initial state S
- a goal state G,

and returns a list of actions from S to G,

Hint:

```
(defstruct cell
    up-cell
    down-cell
    left-cell
    right-cell
    visited-flag)
```