### CMPUT 325 - Prolog Examples 3

**Specification of Prolog predicates:**

* A parameter marked by **+** is an **input parameter**: it is bound to a value when the predicate is called.
* A parameter marked by a **-** is an **output parameter**: it is a variable when the predicate is called and bound to a value when the predicate is proved.
* A parameter marked by a **?** can be used as either input or output parameter.

#### Problem 1:

Define the predicate

sqsum(+L, ?S)

where L is a given list of integers, and S is the sum of the squares of all numbers in L. Examples:

?- sqsum([1,2,3],X).

should return X = 14. (1\*1 + 2\*2 + 3\*3)

?- sqsum([1,4,2,3],30).

should return yes.

?- sqsum([1,2,3,4,5],10).

should return no.

* [Sample solution](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/sqsum.pl)

#### Problem 2:

Define the predicate

lastN(+L, +N, ?R)

where L is a given list and N is a given integer. You can assume that N is at least 0, and not larger than the length of L. R should be a list which contains the last N items in L. Examples:

?- lastN([7,3,4], 1, R).

should return R = [4].

?- lastN([7,3,4], 3, R).

should return R = [7,3,4].

?- lastN([7,3,4], 0, R).

should return R = [].

?- lastN([7,3,4], 2, [3,4]).

should return yes.

* [Sample solution](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/lastn.pl)

#### Problem 3:

Define the predicate

translate(+D, ?W, ?T)

where D is a dictionary, W is a list of words in one language, and T is the corresponding list of translated words. Here, a dictionary is represented as a list of pairs (x,y), where x is a word in the first language and y is its translation in the second language. Your function should work in both directions: if W is given, compute T. If T is given, compute W. If both W and T are given, check if the translation is correct. (Hint: in Prolog, you can do all three things with the same set of predicates.) If a word in W or T is not found in the dictionary, its translation should be the same as the given word. See the word "tiger" in the examples.

Examples:

?- D = [(mouse, maus), (horse, pferd), (dog, hund), (fish, fisch), (cat, katze)],

translate(D, [dog, cat, tiger, mouse], T).

should compute T = [hund, katze, tiger, maus].

?- D = [(mouse, maus), (horse, pferd), (dog, hund), (fish, fisch), (cat, katze)],

translate(D, W, [hund, katze, tiger, maus]).

should compute W = [dog, cat, tiger, mouse].

* [Sample solution](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/translate.pl)

#### Problem 4:

You are running a bank that deals in the three currencies Dollar, Euro and Yen. Define the predicate

countmoney(+L, ?D, ?E, ?Y)

where L is a given list containing amounts of money. L may be an empty list. The variables D, E, and Y contain the total amount of money for the three currencies in L respectively. All money is represented in the form of the 1-ary function symbols dollar(Amount), euro(Amount), and yen(Amount). Amounts are integers. Examples:

?- L = [yen(300), euro(20), dollar(1), dollar(7), euro(16), yen(5)],

countmoney(L, D, E, Y).

should return D = dollar(8), E = euro(36), Y = yen(305).

?- countmoney([euro(20), dollar(1), dollar(7), yen(5)],

dollar(8), euro(20), yen(5)).

should return yes, because all the total amounts given are correct.

?- countmoney([euro(20), dollar(1), dollar(7)], dollar(8), E, yen(0)).

should return E = euro(20). It should compute the Euro amount

and verify that the given dollar and yen amounts are correct.

?- countmoney([euro(20), dollar(1), dollar(7), yen(5)], dollar(8), euro(20), yen(6)).

should return no, because the amount of yen does not add up to 6.

* [Sample solution](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/countmoney.pl)

#### Problem 5:

Define the predicate

mostmoney(+L, +ERate, +YRate, -Most)

where L is a list of money as in Problem 4. You can assume that L is not empty. ERate is the exchange rate for Euros (how many cents for 1 Euro), and YRate is the exchange rate (how many cents for 1 Yen). Of course, 1 Dollar is 100 cents. Compute which single amount in L is the most valuable, by using the exchange rates, and return that value in the variable Most. Example:

?- mostmoney([yen(300), euro(20), dollar(1), dollar(7), euro(16), yen(5)], 93, 3, Most).

should return Most = euro(20), since 20 Euro is the largest amount of money in the

list, according to the exchange rates.

In case of a tie, return the leftmost element of L that has the highest value. Example:

?- mostmoney([yen(300), euro(20), dollar(15), dollar(7), euro(16), yen(5)], 70, 5, Most).

should return Most = yen(300).

yen(300) and dollar(15) have the same value, but yen(300) comes first in the list.

* [Sample solution](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/mostmoney.pl)

#### Problem 6 and 7: Computing the distance between cities

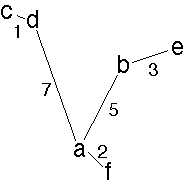
You are given a map with cities connected by roads. The map has no cycles: there is exactly one direct way to go from one city to another. All cities are reachable from each other by roads. (The road network has a tree structure.) Your task is to compute the distance between cities. Cities and roads are represented by a database containing the following predicates:

city(A): A is a city.

road(A,B,L): there is a road between cities A and B of length L.

L is an integer at least 1.

A database will be loaded before your program is run. An example of a database is:



city(a).

city(b).

city(c).

city(d).

city(e).

city(f).

road(a,b,5).

road(a,d,7).

road(b,e,3).

road(f,a,2).

road(c,d,1).

#### Problem 6

Define the predicate

dist(+A, +B, ?D)

where A and B are given cities, and D is the distance between A and B. Examples:

?- dist(b,a,X).

should return X = 5.

?- dist(d,d,X).

should return X = 0.

?- dist(e,c,X).

should return X = 16.

?- dist(e,c,16).

should return yes.

?- dist(e,c,5).

should return no.

Hints:   
- Each road can be used in both directions, e.g. with a road(a,b,5) you can walk from a to b and from b to a. This can be expressed by a general Prolog rule.   
- Be careful not to go back to the same node that you just came from, it can lead to an infinite loop such as a-b-a-b-a-b-... Because there are no loops in the road network, it helps to avoid other kinds of infinite loops here.

#### Problem 7

Define the predicate

distCities(-A, -B, ?D)

Now, given a distance D, find a pair of cities A, B such that the distance between them is exactly D. And given three variables for A, B and D, generate all pairs of cities and their distances. Your function should be able to generate all solutions by backtracking. Your solutions should be generated in the following order: First, city A should appear in the solutions in the same order as in the city database. Second, within all solutions with the same city A, city B should appear in the same order as in the database. Hint: You can use a query such as city(X) to generate all cities in the correct order by backtracking.

Examples:

?- distCities(A, B, 8).

for the database above, this should return the following results in the following order:

A = a, B = c;

A = a, B = e;

A = c, B = a;

A = e, B = a;

?- distCities(A, B, 16).

A = c, B = e;

A = e, B = c;

?- distCities(A, B, 4).

no

?- distCities(A, B, X).

A = a, B = a, X = 0;

A = a, B = b, X = 5;

A = a, B = c, X = 8;

A = a, B = d, X = 7;

A = a, B = e, X = 8;

A = a, B = f, X = 2;

A = b, B = a, X = 5;

A = b, B = b, X = 0;

etc...

Hint: For testing your program, you can use the builtin Prolog predicate *findall* to do the backtracking for you and collect all solutions in one list.

Examples using the database above:

?- findall((A,B), distCities(A, B, 8), L).

will return a list with all solutions and put them in the form (A,B)

L = [(a,c), (a,e), (c,a), (e,a)]

?- findall([A,B,X], distCities(A, B, X), L).

returns a list of 3-element lists with all 36 distances,

L = [[a,a,0],[a,b,5],[a,c,8],[a,d,7],...,[f,d,9],[f,e,10],[f,f,0]]

* [Sample solution for problems 6 and 7](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/dist.pl)

#### Problem 8

Implement the Quicksort algorithm with predicate

quick\_sort(+List,?Sorted)

Example:

?- quick\_sort([4,2,1,3],Sorted).

Sorted = [1, 2, 3, 4] .

* [Two different sample solutions, and a test predicate](https://webdocs.cs.ualberta.ca/~mmueller/courses/325-general/code/prolog/qsort.pl)

#### Problem 9

Symbolic Differentiation and Algebraic Simplification. See the [code](http://www.j-paine.org/prolog/mathnotes/files/symdiff.pl) by [Jocelyn Ireson-Paine](http://www.j-paine.org/).