Departamento de Ciência de Computadores - FCUP

Logic Programming Exam — PART 1

(Duration: 1h 30min)

Date: January 7th, 2017

1) Write a program that can find a the indice of elements of a sublist of a list, where elements are indexed from 1. For example, the query:

```
?- indice([c,e],[a,b,c,d,e],Indice).
```

Should return: `Indice = [3,5]`

```
indice(Elements,List,Indice) :-
   indice(Elements,List,1,Indice).

indice([],_,_,[]).
indice([E|T1],[E|T2],I,[I|Indxs]) :-
   I1 is I + 1,
   indice(T1,T2,I1,Indxs), !.
indice([E|T1],[Other|T2],I,Indxs) :-
   I1 is I + 1,
   indice([E|T1],T2,I1,Indxs).
```

2) Given a list of N sublists, for example,

```
[[a,b,c],[a,b,d],[a],[a,b,d,e,f]]
```

Write a program that returns a list with all elements and their respective frequencies, in order. In the example above, the result should be: `[[c,1],[e,1],[f,1],[d,2],[b,3],[a,4]]`.

```
:- use_module(library(lists)).

frequencies(List,Freqs) :-
   flatten(List,FlatList),
   freq(FlatList,UnsrtFreqs),
   mysort(UnsrtFreqs,Freqs).

freq([],[]).
freq([H|T],[[H,F]|Freqs]) :-
   count_and_del(H,T,F,NewT),
   freq(NewT,Freqs).

count_and_del(H,T,Freq,NewT) :-
   count_and_del(H,T,1,Freq,NewT).

count_and_del(_H,[],N,N,[]).
count_and_del(H,[H|T],I,N,T1) :-
   I1 is I + 1,
   count_and_del(H,T,I1,N,T1), !.
```

```
count_and_del(H,[Other|T],I,N,[Other|T1]) :-
   count_and_del(H,T,I,N,T1).

mysort([],[]).
mysort([[H,F]|T],Sorted) :-
   partition(T,[H,F],L1,L2),
   mysort(L1,S1),
   mysort(L2,S2),
   append(S1,[[H,F]|S2],Sorted).

partition([],_,[],[]).
partition([[H1,F1]|T1],[H2,F2],[[H1,F1]|L1],L2) :-
   F1 =< F2, !,
   partition(T1,[H2,F2],L1,L2).
partition([[H1,F1]|T1],[H2,F2],L1,[[H1,F1]|L2]) :-
   partition(T1,[H2,F2],L1,L2).
```

3) Write a program that accepts as input a Prolog term that describes an expression and returns a Prolog term that is its simplification. For example, the text `4 * x^2 + 4` should return `4 * (x^2 + 1).` (the result should be a Prolog term, not a string nor a list of ASCII codes).

```
simplify(A*X^E1 + B*X^E2, C*X^E1 ) :-
    E1 = E2,
    C is A + B.
simplify(A*X^E1 + B*X^E2, C*X^E1 * (D + E*X^E3) ) :-
    E1 < E2, !,
    E3 is E2 - E1,
    mdc(A,B,C),
    D is A/C,
    E is E/C.
simplify(A*X^E1 + B*X^E2, C*X^E2 * (D*X^E3 + E) ) :-
    E3 is E1 - E2,
    mdc(A,B,C),
    D is A/C,
    E is E/C.
```

This is just part of a general solution, but I accepted solutions that used something similar, including the use of DCGs. You need to implement a predicate that calculates the mdc / mcd (maximum common divisor) between 2 numbers.

4) The program below implements a mysterious task.

- What do you think this program implements?

  This program counts the number of occurrences of an element X in a list (frequency of X in a list).

- Why do you need a cut in this program?

  Because clauses 2 and 3 of this program are not exclusive. In other words, clauses 2 and 3 will succeed if X belongs to the list OR if X does not belong to the list. If, for any reason, a call to c/3 in clause 2 fails, and there is no cut after that call, clause 3 will immediately succeed, yielding a wrong result.

```
c(_, [], 0).
c(X, [X|L], N):-
        c(X, L, R), !,
        N is R+1.
c(X, [_|L], N):-
        c(X, L, N).
```

This example was in list of exercise number 5!

5) Write a program that can insert an element in an ordered binary tree.

```
lookup(K,tree(K,_L,_R,IK),IK).
lookup(K,tree(Other,L,_R,_IOther),IK) :-
   K @< Other, !,
   lookup(K,L,IK).
lookup(K,tree(_Other,_L,R,_IOther),IK) :-
   lookup(K,R,IK).
```

This example was given and explained in class!

6) Suppose we have the following facts:

```
teaches(dr_fred, history).        studies(alice, english).
teaches(dr_fred, english).        studies(angus, english).
teaches(dr_fred, drama).          studies(amelia, drama).
teaches(dr_fiona, physics).       studies(alex, physics).
```

What is the function of the cut operator in the query:

```
?- teaches(dr_fred, Course), studies(Student, Course), !.
```

The cut operator will prune all solutions after the first solution for Course,Student is found. Prolog executes teaches(dr_fred,Course), and binds the variable Course to the value “history”. It then tries to find a solution for studies(Student,history). As there is no solution to this predicate call, Prolog backtracks (!!! NOTE THAT THE CUT WAS NOT EXECUTED YET, BECAUSE PROLOG BACKTRACKS/FAILS BEFORE REACHING THE CUT !!!). Prolog backtracks, unbinds the variable Course and tries the next alternative for teaches(dr_fred,Course), binding variable Course for the second time with the value “english”. Prolog then tries to prove studies(Student,english). This succeeds (!!!) binding variable Student with value “alice”. At this point, the cut is executed. If you ask for another solution, Prolog will try to backtrack past the cut (which has been already executed). Because the cut was executed, Prolog can not backtrack anymore. The only solution found is $Course = english$ and $Student = alice$.