NAME:

CONTACT PERSON AT K.U.LEUVEN (IN CASE OF ERASMUS STUDENTS):

Examination of Programming languages and Programming Methodologies

10 August 2016, 9h00 – 13h00

General Guidelines

The examination is **closed book** and takes (at most) 4 hours.
Make sure that your handwriting is **readable**.
Write your name on every sheet you hand in. Write on **one** side of the sheet only.
When writing down predicates, add comments that describe their meaning. When using particular data structures, document them.

Questions

1. Do the following Prolog queries succeed or fail? In case of success, give the bindings of the variables in the query. In case of failure, explain why.

   1) `?- [X | T] = [ [Z,4], [b,X] ].`
   2) `?- g(3 + Y, 7) = g(A, B), Y = 4, A = B .`
   3) `?- [ s(s(X)), Y | Z ] = [ A, s(s(B)), 3, 4 ], A = Y.`
   4) `?- (a(3) :- B1, B2) = ( a(Z) :- Z is Z1 +1, a(Z1)).`

Concepts used
→ matching process (pg 40)
→ diff b/n matching of structured objects
2 matching of lists
2 matching of clauses (pg 45)

→ pg 41

Ch-1,2,3,4

Clause / rule
Fact
Ques

Trace

2. Consider the following Prolog program.

```
drop(_E,[],[]).
drop(X,[X|T1],T2) :- drop(X,T1,T2).
drop(X,[H|T1],[H|T2]) :- drop(H,T1,T2).
```

Do the following queries succeed or fail? Sketch their execution by Prolog (e.g. by giving an execution tree). In case of success, indicate how many times it succeeds and give the bindings of the variables (if any) in the query. In case of failure, explain why.

1) `?- drop(1,[2,3,3],P).`
2) `?- drop(1,[1,1],P).`

Constraint

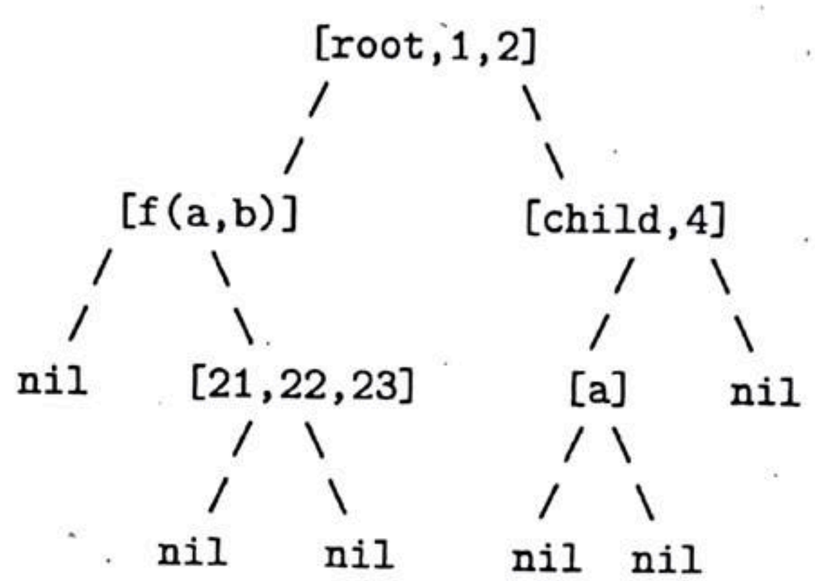
3. The great mezzo-soprano Flora Nebbiacorno has retired from the international opera stage, but she still teaches master classes regularly. At a recent class, her five students were one soprano, one mezzo-soprano, two tenors, and one bass. (The first two voice types are women's, and the last two are men's). Their first names are in random order Chris, J.P., Lee, Pat, and Val – any of which could belong to a man or a woman – and their last names are Kingsley, Robinson, Robinson (the two are unrelated but have the same last name), Ulrich, and Walker. You also know that:

1. The first and second students were, in some order, Pat and the bass.
2. At least one of the second and the third students is a tenor.
3. Kingsley and the fifth student (who isn't named Robinson) were, in some order, a mezzo-soprano and a tenor.
4. Neither the third student, whose name is Robinson, nor Walker has the first name of Chris.
5. Ulrich is not the bass or the mezzo-soprano.
6. Neither Lee or Val (who wasn't third) is a tenor.
7. J.P. wasn't third, and Chris wasn't fifth.
8. The bass isn't named Robinson.

Write a program to determine the order in which these five sang for the class, identifying each by full name and voice type.
Indicate clearly whether you are using CLP or just normal Prolog.

Parsing Tricks

4. Consider a binary tree with nodes whose values are non-empty lists. A tree node is represented by a Prolog term that takes the form `Left-SomeList+Right` or `nil` (note that parentheses can be used to enforce precedence). For example, the following Prolog term would be a valid tree: `(nil-[f(a,b)]+(nil-[21,22,23]+nil))-[root,1,2]+((nil-[a]+nil)-[child,4]+nil)`.
The root node of the example binary tree is `[root,1,2]`, the left child is the binary tree `nil-[f(a,b)]+(nil-[21,22,23]+nil)` and the right child is the binary tree `(nil-[a]+nil)-[child,4]+nil`:



1) Write a predicate `preorder(Tree,ValueList)` that unifies `ValueList` with a list containing all of the tree nodes values from a pre-order tree walk (i.e. emit node value, then left subtree, then right subtree). The predicate should fail if any of the tree nodes values are not of the correct form.
For the above example tree, `preorder/2` unifies `ValueList` with `[[root,1,2],[f(a,b)],[21,22,23],[child,4],[a]]`. You may assume that the argument `Tree` is given (i.e. is input) when `preorder/2` is called and that `append/3` has been defined already.

2). Now write a predicate `preorder_dl/2` that behaves exactly like your `preorder/2` predicate, but in its implementation makes use of difference lists, instead of using `append/3`.

5. The local school has a number of busses. Each bus makes a tour: it stops at a sequence of stops and picks up all the children assigned to a stop. Your Prolog predicates are intended to help with the planning of these tours.

   Some practicalities:

   - All the stops have names given by Prolog atoms: examples are `heverlee`, `leuven-station`, `leuven-city-center`, `a`, `b`, `c`.

   - The distances between the stops are given by Prolog facts for the predicate `distance/3`, for example:

     ```
     distance(heverlee,leuven-city-center,10).
     distance(leuven-station,leuven-city-center,8).
     distance(a,b,2).
     distance(c,b,4).
     ....
     ```

   - The number of children to be picked up at the stops is given by Prolog facts:

     ```
     stop(heverlee,4).
     stop(leuven-station,10).
     stop(leuven-city-center,10).
     stop(a,1).
     stop(c,1).
     ```

     If a stop is not mentioned as a Prolog `stop/2` fact, you may assume no children have to be picked up at that stop (for example, as stop `b` is not in the above list, it implies that no stop is needed there).

Knowing this,

1) Define a data structure to represent the tour of a bus, namely the sequence of the stops of the bus. Use this representation to represent that a bus stops first at `leuven-station` then at `leuven-city-center` and finally in `heverlee`.
2) Write a predicate `costBus` that for a given tour determines the cost of the tour. We use a simple cost function: we add up the distances between the stops of the tour. in the previous example this would give: $8 + 10 = 18$. (we do not take into account the distance of the school to the first and last stop).