Name:
Student number:

Warning : this is just an example of an exam for PLPM. It will be discussed with the students in more detail.

Examination of **Programming languages and Programming Methodologies** example exam, 2019-2020

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.

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.

   (a) `?- [ A, B ] = [ [ 2, 3 ] | C]`

   (b) `?- [ 1, A, a ] = [ B, 2 | 5 ]`

   (c) `?- X = 3 + Y, X is 4 .`

   (d) `?- X = [ 3, 4 | Y], Y = Z, Z = [ 2 | T ].`

   Alternative kind of question:
   What is the domain of the decision variable Z in CLP(FD) after the following query:

   ```
   ?- X in 1..7, Y in -3..100, Y #> X, Z #\= 0,  Z #= Y - X.
   ```

2. Consider the following Prolog predicate `compress/2`.

```
compress([ ], [ ]) .
compress([ X ], [ X ]) .
compress([ X, X | Xs ], Zs) :- !, compress([ X | Xs ], Zs ).
compress([ X, Y | Xs ], [X | Zs]) :- compress([ Y | Xs ], Zs ).
```

   Does the query `?- compress([3,4,4,3], L).` succeed or fail? Sketch its 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.

   Alternative question:
   Does the query `?- compress([4,4,4,4], [4,4]).` succeed or fail? Sketch its 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.

   Remark: if a query has more than 1 solution, you should give the complete execution trace with all the solutions.

3. **Remark: this exercise is typically similar to one the students have solved during the course or the exercise sessions. It does not have to be about CLP programming.**.
   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 whether you are using CLP or just normal Prolog. Also indicate where in your code you are dealing with each of the hints.

4. After the exams of January, Eveline has the scores of the students for the courses. For example, student Danny obtained a 20 on FAI and a 15 on PLPM, Jonas obtained a 18 on PLPM, a 14 on FAI, and a 4 on UAI.

   (a) Represent this information in two different ways: by Prolog facts and by Prolog terms. Use your 2 representations to represent the above example.
   (b) Define for both representations the predicate `topscore` that determines for a given course the student with the top score. You may assume that for each course there is exactly 1 student with a top score.
   In the above example, Danny is the student with the top score for FAI, whereas for UAI it is Jonas.

5. **First example of a question 5:**

   Consider a set of dominoes. Each domino is a rectangle consisting of two parts and each part has a number of spots on it. The number of spots can go from 0 to 6. There are 28 different dominoes. Using four dominoes we can make a configuration as in Figure 1. Note that we used 4 different dominoes and that for each row and column the sum of the spots on the relevant parts of the dominoes is exactly 3, except for the center row and column.

   (a) Define a data structure that represents a domino and a data structure that represents such a configuration of 4 dominoes.
   (b) Write a predicate that checks whether a given configuration is a good one.
   (c) Write a predicate that succeeds when two given configurations are variants. Note that if we turn the configuration of Figure 1 over for example 180 degrees we get its variant in Figure 2. These

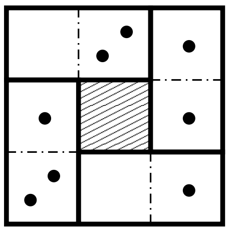

Figure 1: A configuration of dominoes

kinds of variants should be recognised. Note that you can also turn over 90 and 270 degrees.

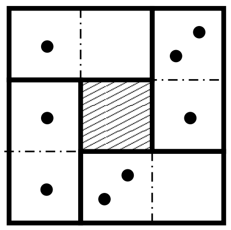

Figure 2: A variant configuration of dominoes

(d) Write a predicate that searches all possible good configurations of 4 dominoes, that eliminates the variants and that prints the configurations out in the form

0 2 1 1 1 0 2 1

(i.e. giving the number of spots on the parts of the dominoes starting in the up ward left corner and going clockwise).

Second example of a question 5:

There are several compression techniques to represent data in a compact manner. In this case, we are working in a setting where every character corresponds with a *code* for this character. A code is a sequence of `0`s and `1`s. Our compression technique has one important property: no code is the prefix of another code.

If the character k has `0` as its code, then all codes for other characters must start with a `1` (`1 ...`) to prevent ambiguity with the code for k.

To keep things simple we assume that for a given N we are encoding the characters *0*, *1*, *2*, ..., *(N-1)*.

The problem is that we know N and we know the encoding for the character sequence *0 1 2 3 ... (N-1)*, but we lost the codes for the individual characters. So we need your help to reconstruct the individual codes.

(a) How do you represent a code (for a character)? How do you represent the code `1 1 0 1 0` ?

(b) Take as example the codes for 3 characters, namely the characters *0*, *1*, *2*. How do you represent the following information(actually the encoding scheme): character *0* has code `1 1`, character *1* has code `0` and character *2* has code `1 0`?

(c) Write a predicate `prefix(Codes, Prefix, RestCodes)` which succeeds if `Prefix` is a prefix of `Codes` and `RestCodes` is `Codes` without the prefix. Note that the prefix is intended to be used as a code, so it should contain at least one `0` or one `1`.

(d) Write a predicate `acceptableCode(Code,CodesSoFar)` that given `Code` and `CodesSoFar` succeeds if no code is the prefix of another code. The codes in `CodesSoFar` are not prefixes of each other. The predicate `acceptableCode` succeeds for the code `1 1` if `CodesSoFar` consists of the codes `0 1` and `0 0 1`.
The predicate `acceptableCode` fails for the code `0 0 1` if `CodesSoFar` consists of the codes `0 1` and `0 0`.

(e) Write a predicate `decode(N, Codes, Res)`. This predicate takes as input `N`, the number of characters, and `Codes`, the encoding for the character sequence *0 1 2 3 ... (N-1)*. The codes in `Codes` are **ordered**. This means that the code for character *0* is at the start of the list, followed by the code for character *1* etc. Your task is to unify `Res` with a possible encoding scheme (see subquestion 2)). For this you must take into account the prefix constraint explained above.

Example 1. Assume that `N` is 3 and `Codes` is `1 1 0 1 0`, then `decode` succeeds with `Res` respresenting the information that character *0* has code `1 1`, character *1* has code `0` and character *2* has code `1 0`.
If the character *0* would be assigned the code `1`, there is no valid

code for the character *1*, because those would all start with `1` (`1 0` or `1 0 1` or ...). By taking `1 1` as code for *0*, this problem does not occur.

Example 2. Assume that `N` is 4 and `Codes` is `0 0 0 1 1 1 0 1 0` , then `decode` succeeds twice. In the first answer `Res` respresents the information that character *0* has code `0 0`, character *1* has code `0 1`, character *2* has code `1 1 0`, and character *3* has code `1 0`. In the second answer `Res` respresents the information that character *0* has code `0 0`, character *1* has code `0 1 1`, character *2* has code `1`, and character *3* has code `0 1 0`.

Good luck. Gerda Janssens