Final Examination, CMPUT 325

Dec 9, 2003

Last name: _____

First name: _____

time(hours(3)).
numberofquestions(7).
pages(9).
totalmarks(100).
marks(question1, 18).
marks(question2, 12).
marks(question3, 14).
marks(question4, 16).
marks(question5, 10).
marks(question6, 15).
marks(question7, 15).

2020 students: I removed questions 1,2,6,7,
since they are about topics
that are not on this year's final exam

This is a 'closed book' exam, you cannot use any notes or books or computers etc.

Write answers in the *space directly after each question* (preferred), or make it *very* clear what the answer for each question is, if you write it on the left side page. On the exam pages (right pages), cross out everything that you wrote that should not be part of your answer.

3 Unification (14 marks total)
What is the result of the following queries in Prolog?
1. Give the overall outcome.
Circle one of: **yes** if query succeeds, **no** if query fails, or **error** if there is an error in execution.
2. Give the variable bindings that are returned from the query.
Use new names such as _1, _2, _3,... for newly created variables, if necessary.
Assume that no user-defined program has been loaded before executing your queries.

Example: **?- Z=0.** outcome: yes bindings: Z=0

3.1 (5x1 mark)

?- X = 3*5.
outcome: yes no error bindings:

?- X = 3*5, Y is X.
outcome: yes no error bindings:

?- Y is X, X = 3*5.
outcome: yes no error bindings:

?- X = 5, Y = X.
outcome: yes no error bindings:

?- X = 5, Y == X.
outcome: yes no error bindings:

3.2 (3x2 marks)

?- p(X, X) = p(f(3, Z), f(Y, Y)).
outcome: yes no error bindings:

?- p(X, f(b, Y)) = p(g(Y, Z), f(Z, a)).
outcome: yes no error bindings:

?- p(f(X), Y, f(f(Z))) = p(Z, f(Z), f(Y)).
outcome: yes no error bindings:

3.3 Most general unifier (3 marks)
For the following terms **t1** and **t2**, find the most general unifier **w**, if it exists, or otherwise show that the terms cannot be unified. Show all the steps in the unification procedure.

t1 = f(f(X, Y), a) **t2 = f(f(f(Y, 3), a), Y)**

4 Understanding a Prolog Program and Backtracking (16 marks)

```
vowel(a).
vowel(e).
vowel(i).
consonant(b).
consonant(c).

letters([C]) :- consonant(C).
letters([C,V|R]) :- consonant(C), vowel(V), letters(R).

fiveletters1(W) :- letters(W), length(W,5).

fiveletters2(W) :- W = [C1,V1,C2,V2,C3],
            consonant(C1), vowel(V1), consonant(C2),
            vowel(V2), consonant(C3).
```

4.1 (3x3 marks) Given the Prolog program above, what is the first answer for the following Prolog queries, and what are the results of asking for more answers by pressing semicolon **;**

?- `letters(X).`

first:

next four:

?- `fiveletters1(X).`

first:

next four:

?- `fiveletters2(X).`

first:

next four:

4.2 (3 marks) Do the two queries **`fiveletters1(X)`** and **`fiveletters2(X)`** compute the same answers with backtracking? If yes, why? If no, why not?

4.3 (2x2 marks) One way to find out how many answers in total are computed by **`fiveletters2(X)`** is to issue the query

`?- findall(X,fiveletters2(X),L),length(L,N).`

4.3.1 What is the value of N computed by this query?

4.3.2 How many *different* answers are contained in L in the result of this query?

5. Questions on Constraint Programming (10 marks total)

Are the following statements true or false? Circle **true** or **false** only if you know the answer. Do *not* make a blind guess if you do not know the answer. Giving *more than one* wrong answer will *reduce* your marks according to the formula: marks = right - (wrong - 1).
For example, 5 right 2 wrong = 5 - 1 = 4 marks.
In the questions, CLP stands for "constraint logic program" or "constraint logic programming".

true false In principle, constraint logic problems could also be solved by using Lisp.

true false CLP can be much more efficient than Prolog backtracking.

true false Prolog's backtracking mechanism is useful for a CLP solver.

true false CLP is a more general problem-solving method than logic programming.

true false A variable domain in CLPFD must be a range of consecutive integers.

true false A problem with two contradictory constraints can still have a solution.

true false Both Prolog and CLP can handle constraints involving uninitialized variables.

true false All variables in a CLP have the same domain.

true false A problem with no constraints on a variable can have a solution.

true false Arc consistency uses constraints over three variables.