Final Examination, CMPUT 325

April 18, 2002
Section B2, instructor Martin Müller

Last name: _____

First name: _____

time(minutes(120)).
numberofquestions(4).
pages(10).
totalmarks(100).
marks(question1, 25).
marks(question2, 30).
marks(question3, 20).
marks(question4, 25).

```
2020 students: I removed questions 1,4,
      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.

This exam is printed on the right side pages only, so you have some space on the empty left pages for temporary notes.
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.

This is a long exam. Maybe you cannot finish all of it - but most of the questions have very short answers, so use your time wisely.

All programming examples have short solutions, which may include writing some small helper functions. So before you spend a lot of time on a complicated solution, think about the problem for a little bit longer.

For writing code, the correctness and clarity of your code are both important. You don't need to write comments or test cases for your code in this exam.

Use only standard functions and predicates that we talked about in class.
Use names starting with an underscore such as _1, _2, _3,... for newly created variables in Prolog.

For true-or-false questions, circle either **true** or **false. Do not make a blind guess if you don't know the answer. Giving too many wrong answers will reduce your marks.**

Examples:
0.0.0 (true) false Today is the final exam for 325 (circle the right answer)

0.0.1 true false It will snow on April 18, 2003 (you don't know, so do not circle anything)

1.3 Higher-order and Lambda Functions (6 marks)

Given the following definitions of **f** and **s**, determine the result of the expressions below.

```
(defun f (x y) (cons x y))
(defun s (f) (function (lambda (x y) (funcall f y x))))
```

1.3.1 (2 marks)

(mapcar 'f '((a) (b)) '((1) (2)))

1.3.2 (2 marks)

(mapcar (s 'f) '((a) (b)) '((1) (2)))

1.3.3 (2 marks) This question uses the standard function **append**.

(mapcar (s 'append) '((a) (b)) '((1) (2)))

2. Prolog (30 marks total)

2.1 Unification (8 marks)
What is the first result of the following queries in Prolog? Give the overall outcome (circle one of: **yes** if query succeeds, **no** if query fails, **infinite-loop**, or **other-error**) as well as the variable bindings. Use 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

2.1.1 (2 marks)

?- p(X) = p(5).

outcome: yes no infinite-loop other-error

bindings:

2.1.2 (2 marks)

?- X = 20, f(Y) = f(g(Z)), Z = g(T).

outcome: yes no infinite-loop other-error

bindings:

2.1.3 (2 marks)

?- X is 5+3, Y=X+X.

outcome: yes no infinite-loop other-error

bindings:

2.1.4 (2 marks)

?- X = 5+3, Y is X+Z.

outcome: yes no infinite-loop other-error

bindings:

2.2 Understanding a Prolog Program and Backtracking (8 marks)
Consider the following Prolog program:

```
tag(1).
tag(2).
label(a).
label(b).
label(c).

p([]).
p([T,L|R]) :- tag(T), label(L), p(R).
q([L|R]) :- label(L), p(R).
r(X) :- p(X).
r(X) :- q(X).
```

What is the first result for the following Prolog query, and what are the results of asking for four more answers by pressing semicolon **;** four times?

2.2.1 (2 marks)
?- p(X).

first:

next four:

2.2.2 (2 marks)
?- q(X).

first:

next four:

2.2.3 (2 marks)
?- r(X).

first:

next four:

2.2.4 (2 marks) Assume that in the program above, the order of the two clauses for r(X) is swapped:
r(X) :- q(X).
r(X) :- p(X).
Now what are the results of the query **?- r(X).**

first:

next four:

2.3 Understanding and Changing Prolog Programs (14 marks)

Consider the following Prolog program:

```
member(A,[A|_]).
member(A,[B|L]) :- A \== B, member(A,L).
a(X,L,L) :- member(X,L), !.
a(X,L,[X|L]).
```

2.3.1 (6x1 mark)
Determine the first answer for the following queries, and give the bindings for the variables.
If a query fails, explain why.

2.3.1.1 **?- a(3, [1,2], X).**

2.3.1.2 **?- a(3, [3,1,2], X).**

2.3.1.3 **?- a(3, [1,2,3,3], X).**

2.3.1.4 **?- a(3, [1,[3],2], X).**

2.3.1.5 **?- a(3, X, X).**

2.3.1.6 **?- a(3, X, [2|Y]).**

2.3.2 (5 marks)
Write a Prolog predicate **a1** that computes exactly the same first answer as **a** but does *not* use the cut. You must code all the helper functions that you may want to use, except for **member**.

2.3.3 (1 mark)
What about the efficiency of the two predicates? Circle one of the three given answers.
a) Predicate **a** is more efficient.
a1) Predicate **a1** is more efficient. **a** **a1** **same**
same) Both are about the same.

2.3.4 (2x1 mark) answer true or false.

2.3.4.1 true false Predicate **a** can generate more than one answer with backtracking.
2.3.4.2 true false Predicate **a1** can generate more than one answer with backtracking.

3. Constraint Programming (20 marks total)
3.1 Completing a Constraint Program (10 marks)

Professor C.S.Train had to schedule five meetings on the same day, and solved this problem by using Sicstus Prolog and the CLPFD package.
She used the following 5 variables to represent the starting times of meetings:
S1 .. meet student1 (duration: 1 hour)
S2 .. meet student2 (duration: 2 hours)
R .. research group meeting (duration: 2 hours)
L .. lunch with colleagues (duration: 1 hour)
B .. breakfast with visitor (duration: 1 hour)

Here are the constraints:
- all meetings start at full hours (e.g. 8:00, but not 8:30)
- no meetings before 8:00
- all meetings must end at 19:00 at the latest
- order: breakfast first, meet student1 before lunch, student2 after lunch, research meeting last.
- lunch should start either at 11:00, 12:00, or 13:00.

3.1.1 (2x1 mark)
Compute two different solutions to the constraint problem by hand.

Solution 1: S1 = S2 = R = L = B =

Solution 2: S1 = S2 = R = L = B =

3.1.2 (8 marks)
Here is Professor Train's program, with some parts left out. Write the missing parts in the space provided after the program.

```
MISSING-PART-1    % load CLPFD package

todaysmeetings(M) :-
      M = [S1,S2,R,L,B],      % list of all meetings
      MISSING-PART-2,    % define variables and domains
      constrain(M),             % check given constraints
      MISSING-PART-3.    % solve constraint problem (use default settings)

constrain(M) :-
      MISSING-PART-4.    % check given constraints
```

Write the missing parts here:
3.1.2.1 (1 mark) MISSING-PART-1

3.1.2.2 (2 marks) MISSING-PART-2

3.1.2.3 (1 mark) MISSING-PART-3

3.1.2.4 (4 marks) MISSING-PART-4

3.2 Questions on Constraint Programming (10x1 mark)

Are the following statements true or false? Circle **true** or **false** only if you know the answer. In the questions, CLP stands for "constraint logic program" or "constraint logic programming".

3.2.1	true	false	Prolog's backtracking mechanism is useful for a CLP solver.
3.2.2	true	false	CLP is a more general problem-solving method than logic programming.
3.2.3	true	false	All variables in a constraint problem have the same domain.
3.2.4	true	false	A variable domain in CLPFD must be a range of consecutive integers.
3.2.5	true	false	In principle, constraint problems could be solved by using C or Java.
3.2.6	true	false	Current CLP systems are useful for industrial applications.
3.2.7	true	false	A problem with two contradictory constraints can still have a solution.
3.2.8	true	false	CLP is designed to replace all previous forms of logic programming.
3.2.9	true	false	A CLP can be 1 million times more efficient than Prolog backtracking.
3.2.10	true	false	CLPFD contains one fixed strategy for assigning values to variables.