Final Examination, CMPUT 325

April 24, 2001
Section B1, instructor Martin Müller

Last name: ______________________

First name: ______________________

time(minutes(120)).
numberofquestions(6).
pages(10).
totalmarks(100).
marks(question1, 18).
marks(question2, 12).
marks(question3, 15).
marks(question4, 20).
marks(question5, 15).
marks(question6, 20).

This is a 'closed book' exam, you cannot use 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 some 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. Before you spend a lot of time on a complicated solution, it could be a good idea to think about the problem for a little bit longer.

For writing Lisp and Prolog code, the correctness and clarity of your code are both important, but 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.

Student ID ___________

Question 1 (18 marks) _________
Question 2 (12 marks) _________
Question 3 (15 marks) _________
Question 4 (20 marks) _________
Question 5 (15 marks) _________
Question 6 (20 marks) _________

Total (100 marks) _________

1. Writing Lisp Code (18 marks total)

1.1 (6 marks) An integer n is called a cube if there is another integer k such that $n = k^3$. Define a Lisp function **cubesum**, which takes a list of integers as input and computes the sum of cubes of list elements. That is, for a list (a_1 a_2 ... a_n) compute $a_1^3 + a_2^3 + ... + a_n^3$. **cubesum** should return 0 for an empty input list.

1.2 (6 marks) Define a Lisp function **iscube** which takes an integer n as an argument and returns T if n is a cube and nil otherwise.
Remark 1: Your function should be correct, but it does not have to be efficient. Do not use any special mathematical functions such as logarithms for this problem.
Remark 2: Negative numbers can be cubes, too. For example, $-8 = (-2)^3$.

1.3 (6 marks) Define a Lisp function **nocubes**, which takes a list of integers as input and returns the list elements which are **not** cubes, in the same order as the input.
Example: for the list (1 5 0 -9 -8 12 27 13), the correct answer is (5 -9 12 13). The numbers 1, 0, -8 and 27 are cubes and are therefore not contained in the answer.

2. SECD machine (12 marks total) Consider the following Lisp expression:

(if (atom 9) (+ 5 (* 6 2)) (* 3 5))

2.1 (5 marks) Compile this expression into SECD code.

2.2 (7 marks) Show the execution of the compiled code on the SECD machine. Show the s,e,c,d stacks after each execution step.
Reminder: unlike Prolog, the Lisp function **atom** evaluates to **T** for numbers.

3. Unification (15 marks total)

3.1 (1 mark) Is the following statement true or false? Why?

> "Two terms have either no unifier or exactly one unifier."

3.2 (2x2 marks) For each pair of atoms below, determine whether they are unifiable. If yes, show the most general unifier. In these examples, X, Y and Z are the variables.

3.2.1 q(f(X), g(f(Y)), f(Z))
q(f(f(Y)), g(f(a)), f(g(Y)))

3.2.2 p(g(f(X)), h(f(f(Z)), Y))
p(g(Y), h(Y, f(X)))

3.3 (5x2 marks)
Consider the following Prolog program:

```
p(X, [X]).
q(X, Y) :- p(Y, X).
```

What is the result of executing the following Prolog queries?
Use names such as _1, _2, _3,... for newly created variables, if necessary.

3.3.1 ?- q(X, a).

3.3.2 ?- q(X, Y).

3.3.3 ?- q(f([Y]), f(X)).

3.3.4 ?- q([f(Y)], f(g(X))).

3.3.5 ?- X = [a | Y], r(X) = r([Z, b]).

4. Understanding Prolog Programs (20 marks total)

Consider the following Prolog program:

```
s(a(X, Y), Z) :- s(X, X1), s(Y, Y1), Z is X1 + Y1.
s(m(X, Y), Z) :- s(X, X1), s(Y, Y1), Z is X1 * Y1.
s(X, X) :- number(X).
```

4.1 (7x1 mark)
Determine the result of the following queries.
If the result of a query is 'no', or if a query leads to an infinite loop, explain why.

4.1.1 ?- s(5, X).

4.1.2 ?- s(a(5, 20), X).

4.1.3 ?- s(a(Y, 20), X).

4.1.4 ?- Y = 20, s(m(5, a(Y, 5)), X).

4.1.5 ?- s(a(5, m(6, 3), 20), X).

4.1.6 ?- Y = 5, Z = 1, s(a(a(Z, m(6, 3)), m(Z, Y)), X).

4.1.7 ?- Y=5, s(a(a(Z, m(6, 3)), m(Z, Y)), X), Z = 1.

4.2 (7x1 mark)
The following program for **s2** is almost the same as the program for **s** in 4.1, but "is" was replaced by "=".

```
s2(a(X, Y), Z) :- s2(X, X1), s2(Y, Y1), Z = X1 + Y1.
s2(m(X, Y), Z) :- s2(X, X1), s2(Y, Y1), Z = X1 * Y1.
s2(X, X) :- number(X).
```

What is the result of trying the same queries as above with **s2** instead of **s**?

4.2.1 ?- s2(5, X).

4.2.2 ?- s2(a(5, 20), X).

4.2.3 ?- s2(a(Y, 20), X).

4.2.4 ?- Y = 20, s2(m(5, a(Y, 5)), X).

4.2.5 ?- s2(a(5, m(6, 3), 20), X).

4.2.6 ?- Y = 5, Z = 1, s2(a(a(Z, m(6, 3)), m(Z, Y)), X).

4.2.7 ?- Y = 5, s2(a(a(Z, m(6, 3)), m(Z, Y)), X), Z = 1.

4.3 (3 marks) Change the program for **s2** from question 4.2 in a way that it:
1. computes the same results whenever the original program terminates.
2. never gets into an infinite loop.
What is the result of your revised program in those cases where the original program does not terminate?
You don't need to write down the whole new program, just specify where and how to change it.

4.4 (3 marks) What happens if you change the program for s from question 4.1 in the same way that you changed the program for **s2** in question 4.3? What are the results?

5. Writing Prolog Code (15 marks total)

A number N is called a prime number if it cannot be divided without a reminder by numbers other than 1 and N.

5.1 (5 marks) Write a Prolog predicate **isprime(N)** that succeeds if and only if a given integer N is a prime number. For example, isprime(17) should succeed but isprime(15) should fail. You can assume that N>1.
Hint: given integers A and B, the reminder in integer division can be computed in Prolog by A mod B.

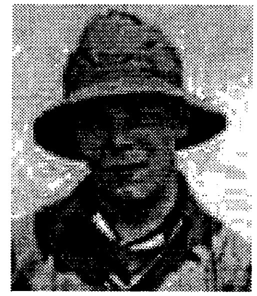
5.2 (5 marks) Write a Prolog predicate **primes(A, B, P)** that computes P, the sorted list of all primes in the range from A to B. A and B are given integers with A>1 and B >= A.
Example: primes(2, 20, P) should return P = [2,3,5,7,11,13,17,19].

5.3 (5 marks) A prime pair is a pair of prime numbers N, M with M = N+2. Write a Prolog predicate **primepairs(A, B, PP)** that computes PP, the sorted list of all prime pairs in the range from A to B. A and B are given integers with A>1 and B >= A. In the list PP, a prime pair N, M should be represented by the structure **pair(N, M)**.
Example: primepairs(2, 30, PP) should return PP = [pair(3,5),pair(5,7),pair(11,13),pair(17,19)].
Note that pair(29,31) is not included in the result, since 31>30.

6. Herbrand (20 marks total)

Let P be the following set of Horn clauses. Let the set of constants be {a,b}.

p(a)
q(b)
p(x) → q(s(x))
q(x) → p(s(s(x)))



Jacques Herbrand

6.1 (1 mark) Determine the functions and predicates in P.

6.2 (2x2 marks) Let H be the Herbrand universe of P and let B be the Herbrand base of P. Consider the following expressions:

1) q(s(s(a)))
2) p(a) → q(s(a))
3) b
4) s(s(s(a)))
5) p(s(s(b)))
6) q(s(p(a)))

6.2.1 Which of the expressions 1)-6) are elements of the Herbrand universe H?

6.2.2 Which of the expressions 1)-6) are elements of the Herbrand base B?

6.3 (2x2 marks) Let S = {p(a), q(b), q(s(a)), p(s(s(b)))}.

6.3.1 Is S a Herbrand interpretation? Why? Or why not?

6.3.2 Is S a Herbrand model? Why? Or why not?

6.4 Consider the least model of P.

6.4.1 (2 marks) Which of the expressions 1)-6) in question 6.2 above are elements of the least model of P?

6.4.2 (3 marks) List 6 more elements of the least model of P.

6.5 (3 marks) Describe all the elements of the least model of P (you can use plain English, or a mathematical notation).

6.6 (3 marks) Give an example of a model of P that is not a minimal model.

(end of exam)