UNIVERSITY OF OSLO

Faculty of mathematics and natural sciences

Examination in	INF3110/4110 — Programming languages
Day of examination:	2. december 2003
Examination hours:	14.30 – 17.30

This problem set consists of 7 pages.

Appendices:	None
Permitted aids:	All printed and written

Please make sure that your copy of the problem set is complete before you attempt to answer anything.

First some general advice and remarks:

- The exam set consists of six independent parts. There are many problems and therefore important that you use the time well
- The points on each part indicate how much weight the different parts will count under the evaluation. In total you can obtain 60 points for this exam.
- If the text is unclear or imprecise in any way, you are free to make you own precisions. In that case, formulate these clearly in your answers.
- Focus on giving short and clear explanations.

Good luck!

(Continued on page 2.)

Problem 1 ML (14 points)

1a Types (8 points)

Which type would ML assign to the following functions? Remember that ML tries to give the most general/polymorphic type for each expression. In other words: for each function, what is the most general type for this function? (Se examples below.)

- `fun f1(x) = [x]`
- `fun f2(x) = [42, x, x, x]`
- `fun f3(x,y,z) = (x,z)`
- `fun f4(x,z,b) = if b then x else 10`
- `fun f5(x) = if (x > 0) then x*f5(x-1) else 1`
- `fun f6(g,x) = g(x)`
- `fun f7 g x = g(x)`
- `fun f8(i) = fn j => if (i = j) then i else j`

Examples:

`fun f(x) = (42, "h")`	is assigned the type	`'a -> int * string`.
`fun f(x) = x`	is assigned the type	`'a -> 'a`.

1b Programming (6 points)

Note: Here it is not allows to use the standard library. Everything should be purely functional; one can not use imperative constructions.

(a) Write a function

```
fun double (ll : int list list) : int list list   = ...
```

which takes a list of lists of integers and doubles each integer in this way:

```
double [[1,2,3],[2,3,4]]   gives   [[2,4,6],[4,6,8]]
```

(b) Write a function

```
fun init (seq: ''a list) (lst: ''a list) : bool = ...
```

(Continued on page 3.)

such that `init seq lst` returns `true` if and only if `seq` is the initial part of the list `lst`:

```
init [1,2] [1,2,3]      gives    true
init [1,2] [0,1,2,3]    gives    false
```

Problem 2 Syntax (10 points)

In this problem we will look at different languages over the alphabet {**0**,**1**}.

For **(a)** and **(b)** below: give (1) a regular expression and (2) a non-deterministic automaton which accepts *exactly* the given strings.

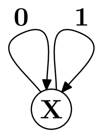
(a) Strings which contains the sequence **010**.

(b) Strings which contains both **0** and **1**.

Example: All strings over {**0**,**1**}.

(1) A regular expression: ⟦**0**|**1**⟧*.

(2) A deterministic automaton:



(c) Transform the non-deterministic automaton from **(a)** to a *deterministic* automaton. Let it be clear (f.ex. from the drawing) how this transformation is done.

Problem 3 Grammars (9 points)

3a (5 points)

Say which alternative are true. (There is at least one correct alternative for each problem, but there can also be more than one. In that case, give exactly the correct alternatives.)

(1) A LL(1)-grammar

(a) can have empty right-hand sides
(b) can be right recursive
(c) can be left recursive

(Continued on page 4.)

(d) none of these match

(2) For every context-free grammar there is

(a) a context-sensitive grammar for the same language
(b) a regular grammar for the same language
(c) a LL(1)-grammar for the same language
(d) none of these

(3) Recursive definition are allowed in

(a) regular grammars
(b) context-free grammars
(c) context-sensitive grammars
(d) LL(1)-grammars

(4) A context-free grammar must be

(a) regular
(b) unambiguous
(c) LL(1)
(d) none of these

(5) The expressive power of classical BNF is

(a) equal to the expressive power if extended BNF
(b) greater than the expressive power if extended BNF
(c) less than to the expressive power if extended BNF

(That **A** has greater xpressive power than **B** here means that **A** can describe all languages that **B** can describe, but not the other way around.)

3b (1 points)

Describe in as few words as possible the language which the following grammar defines.

$\langle S\rangle \rightarrow$ **a** $\langle S\rangle$ **b** | **ab**

3c (1 points)

Describe in as few words as possible the language which the following grammar defines.

$\langle S\rangle \rightarrow$ **a** $\langle S\rangle$ **a** | **b** $\langle S\rangle$ **b** | ε

(Continued on page 5.)

3d (2 points)

Consider the following grammar:

$\langle S\rangle \rightarrow$ **a** $\langle B\rangle$ | **b** $\langle A\rangle$ $\langle B\rangle$
$\langle A\rangle \rightarrow$ **a** | $\langle A\rangle$ $\langle A\rangle$ **b**
$\langle B\rangle \rightarrow$ **b** | **b** $\langle S\rangle$ | **a** $\langle B\rangle$ $\langle B\rangle$

(a) Draw a syntax-tree for the string **baababb**

(b Show that the grammar is ambiguous.

Problem 4 λ-calculus (7 points)

Reduce the following λ-terms as much as possible. Show each reduction step.

1. $(\lambda x.x)z$
2. $((\lambda y.x)(\lambda x.x)x)[x/y]$
3. $(\lambda x.xx)((\lambda b.b)(\lambda c.c))$
4. $(\lambda xy.xyx)(\lambda b.b)(\lambda c.c)$

(Continued on page 6.)

Problem 5 Run-time systems/parameter passing (12 points)

Below is a small program i the language C4” from the course book.

Assume that the procedure P passes its parameters by *reference* and that the procedure Q passes its parameters by *value*.

5a (10 points)

Draw how the data register/“run-time”-stack looks right before Q is called for the *second* time. Make sure to include possible dynamic links, static links, parameters and local variables. Draw this with all values.

```
main()
{
   int x = 0;

   P(int y)       \\ y passed by reference
   {
      Q(int z)    \\ z passed by value
      {
         z := z + 12;
         P(z);
      }

      y := y + 1;
      Q(y);
   }

   P(x);
}
```

5b (2 points)

If Q had passed its parameters by *value-result*, would the data register have appeared different? Why/why not?

(Continued on page 7.)

Problem 6 PROLOG (8 points)

6a Unification (4 points)

This problem is about the PROLOGs unification. Give PROLOGs answer for each of the queries below.

1. `x(G) = x([1,2,3,4]).`
2. `f(a,Z) = f(U, h(U)).`
3. `f(U,h(V)) = f(a,g(a)).`
4. `numbers(1,2,X) = numbers(1,Y,Z).`

Eksempel:

```
?- f(a) = f(X).
```

PROLOG svarer `X = a`.

6b Member (4 points)

Here is the predicate `member`, which finds out which elements that are member of a list.

```
member(E, [E|_]).
member(E, [_|Rest]) :- member(E, Rest).
```

Example of use:

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

Yes
```

Show a complete search tree for the query `?- member(X, [1,2])` and give all the answers PROLOG provides.