Written Examination
DM 509 Programming Languages

Department of Mathematics and Computer Science
University of Southern Denmark

Monday, January 10, 2011, 09:00 – 13:00

This exam set consists of 7 pages (including this front page) and contains a total of 5 problems. Each problem is weighted by the given percentage. The individual questions of a problem are not necessarily weighted equally.

Most questions in a problem can be answered independently from the other questions of the same problem.

All written aids are allowed. Answering questions by reference to material not listed in the course curriculum is not acceptable.

You may answer the exam in English or in Danish.

Problem 1 (20%)

Question a: Implement a Prolog predicate `dropFirst/3` such that `dropFirst(N, L, M)` is true if, and only if, `M` is the list obtained by dropping the first `N` elements of the list `L`.

For example, the query

```
?- dropFirst(2, [6,3,4,5,2,1], M).
```

should yield the answer `M = [4,5,2,1]`. Likewise, the query

```
?- dropFirst(4, [1,2,3], M).
```

should yield the answer `M = []`.

Question b: Implement a Prolog predicate `takeNth/3` such that `takeNth(N, L, M)` is true if, and only if, `M` is the list obtained by taking every `N`-th element from the list `L`.

For example, the query

```
?- takeNth(2, [6,3,4,5,2,1], M).
```

should yield the answer `M = [6,4,2]`. Likewise, the query

```
?- takeNth(3, [1,2,3,4], M).
```

should yield the answer `M = [1,4]`.

Question c: A fraction $\frac{a}{b}$ can be represented by the term `a/b`. Note that instead of "/" one could also use "-" or "+".

Implement a Prolog predicate `add/3` such that `add(X,Y,Z)` is true if, and only if, `Z` is the fraction obtained by adding `X` and `Y`.

For example, the query

```
?- add(1/6, 3/10, F).
```

should yield the answer `F = 28/60`. Likewise, the query

```
?- add(3/1, 1/2, F).
```

should yield the answer `F = 7/2`.

Question d: A heterosquare is a matrix of dimension $n \times n$ containing all numbers from 1 to n^2 such that the sums of all rows and of all columns are pairwise different.

The following is an example of a heterosquare of dimension 2×2. Note that $1{+}2 = 3$, $3{+}4 = 7$, $1{+}3 = 4$, and $2{+}4 = 6$, i.e., we have the sums 3, 7, 4, and 6, which are all pairwise different.

1	2
3	4

We represent such a square as a list of concatenated rows, i.e., the above square would be represented as follows.

```
[1,2,3,4]
```

Implement a Prolog predicate `hetero/1` such that the query `?- hetero(L).` has exactly those lists `L` as answers that represent heterosquares of dimension 2×2.

You may (but do not have to) use constraint logic programming for your implementation.

Problem 2 (25%)

Question a: Consider the following Prolog program.

```
p(X,Y) :- q(X), s(Y), p(Y,X).
p(X,Y) :- r(X), s(Y), p(Y,X).
p(X,2).
q(1).
r(2).
s(3).
s(4).
```

Draw the SLD tree for the query `?- p(A,B).` and list all answers with the instantiations of `A` and `B`.

Question b: We now introduce a cut into the body of the second clause from Question a, i.e., we now have the following Prolog clauses for `p/2`.

```
p(X,Y) :- q(X), s(Y), p(Y,X).
p(X,Y) :- r(X), s(Y), p(Y,X), !.
p(X,2).
```

Indicate in the SLD tree of Question a which branches are cut and list all remaining answers with the instantiations of `A` and `B`.

Question c: For the following pairs of Prolog terms, find a most general unifier or argue that none exists. Show the steps of the algorithm. In case of success, give the resulting substitution. In case of failure, state if it is an occur failure or a clash failure.

1. `p(f(b),a,Y)` and `p(f(Y),X,b)`
2. `q(g(X),g(Y),g(a))` and `q(g(A),A,g(X))`
3. `r(a,Z,[X,Y])` and `r(X,Y,[X|Z])`

Problem 3 (15%)

Question a: Define a HASKELL function `max` which takes two non-negative positive integers and determines the maximum of the two.

For example, `max 2 3 = 3` and `max 5 0 = 5`.

Here, you may not use any pre-defined functions except for `(+)` and `(-)`. In particular, you may not use `(>)` or any other comparison operator.

Question b: Define a HASKELL function `maxList` which takes two lists of non-negative integers of same length and builds a list, which at each position contains the maximum of the elements of the two argument lists.

For example, `maxList [1,6,3] [2,4,5] = [2,6,5]`.

You should use the function `max` from Question a.

Question c: Define a HASKELL function `transpose` which takes a matrix of integers represented as a list of rows and computes the transposed matrix.

For example, `transpose [[1,2],[3,4]] = [[1,3],[2,4]]` and `transpose [[1,2,3,4],[5,6,7,8]] = [[1,5],[2,6],[3,7],[4,8]]`.

Question d: Give a HASKELL declaration for the infinite list `powersOf2` of all strings consisting of just "`*`" with a length that is a power of 2, i.e., a declaration of the form "`powersOf2 = ...`".

For example, `take 5 powersOf2` should return the following list.

```
["*","**","****","********","****************"]
```

for the following (standard definition) of `take`:

```
take 0 _ = []
take _ [] = []
take (n+1) (x:xs) = x : take n xs
```

Problem 4 (20%)

Question a: Consider the following data type for multisets, i.e., for sets that can contain an element multiple times.

```
data MultiSet a = a -> Integer
```

Thus, the expression `\x -> 0` represents the empty multiset $\{\}$ while the expression `\x -> if x == 4 then 1 else if x == 1 then 3 else 0` represents the multiset $\{1, 1, 1, 4\}$.

Functions that work on these multisets need to create, apply, or modify functions. The following declarations for `emptyMS`, `frequency`, and `insert` define the empty multiset, a function returning the multiplicity of an element, and insert an element into a multiset, respectively:

```
emptyMS = \x -> 0
frequency x ms = ms x
insert x ms = \y -> ms y + if x == y then 1 else 0
```

Define a HASKELL function `union` which takes two `MultiSet a` and produces the union of the two multisets.

For example,
`union (insert 1 (insert 1 emptyMS)) (insert 4 (insert 1 emptyMS))`
should return a function representing the multiset $\{1, 1, 1, 4\}$.

Question b: Consider an alternative representation of multisets as lists of pairs where $[(4, 1), (1, 3)]$ represents the multiset $\{1, 1, 1, 4\}$.

Define a HASKELL function `toList` which takes a multiset represented by a value of type `[(a,Integer)]` and returns a list of type `[a]` that contains each element of the multiset as many times as specified.

For example, `toList [(4,1),(1,3)]` could return `[4,1,1,1]` or `[1,1,1,4]`. Note that the order is not important.

Question c: Declare a HASKELL data type `MSList a` using a `data` declaration to represent values of type `[(a,Integer)]` by self-defined data constructors.

Problem 5 (20%)

Question a: Find the most general type for each of the following two HASKELL functions. You may assume that `False,True::Bool`, `1::Int`, and `(:)::a -> [a] -> [a]`.

- `f = \x -> if x 1 then False else True`
- `g (x:xs) y = x (g xs y)`

Explain your reasoning.

Question b: Consider the two following ways of defining HASKELL functions for mapping a function to all elements of a list.

```
map1 f [] = []
map1 f (x:xs) = f x : map1 f xs

map2 f = foldr help [] where
  help x xs = f x : xs

foldr g h [] = h
foldr g h (x:xs) = g x (foldr g h xs)
```

Prove by induction that for all `f` of type `[a -> b]` and `ys` of type `[a]`, these two definitions yield the same result, i.e., `map1 f ys = map2 f ys`.