2012-02-06

2011年度プログラミング言語II（計算機言語I）定期試験問題

1. 型

(1) 次の Standard ML の型に属する式の例を示せ。（各10点）

a) char * string * bool list list * bool

b) ((int * int) * int) list * (real * string list)

(2) 次に示す Standard ML の関数の型を示せ。（10点）

```
fun f(nil, q) = nil
  | f(p::ps, q) = (p * q)::f(ps, q);
```

2. MLプログラミング

以下の関数を Standard ML で実装せよ。

(1) 実数の絶対値を求める関数 abs: real -> real。例えば abs(1.0) の評価値は 1.0、abs(~25.4) の評価値は 25.4 となる。（10点）

(2) 実数のリストから正の要素のみを残したリストを得る関数 positive: real list -> real list。例えば positive([1.0, ~2.5, 3.0, 0.0]) の評価値は [1.0, 3.0] となる。ただし高階関数 simpleMap, reduce, filter などは使わないこと。（15点）

(3) リストを要素とするリストを引数に取り、そのすべての先頭要素からなるリストとそれ以外との組を返す関数 hd_tl: 'a list list -> 'a list * 'a list list。例えば、hd_tl([[1, 2, 3], nil, [4, 5], [6]]) の評価値は ([1, 4, 6], [[2, 3], [5], nil]) となる。（15点）

(4) 文字列のリストの要素をすべて連結した文字列を得る関数 concat: string list -> string を高階関数 simpleMap, reduce, filter のうち一つ以上を用いて実装せよ。例えば concat([“foo”, “bar”, “baz”]) の評価値は “foobarbaz” となる。（15点）

```
fun simpleMap(F, nil) = nil
  | simpleMap(F, x::xs) = F(x)::simpleMap(F, xs);

exception EmptyList;
fun reduce(F, nil) = raise EmptyList
  | reduce(F, [a]) = a
  | reduce(F, x::xs) = F(x, reduce(F, xs));

fun filter(P, nil) = nil
  | filter(P, x::xs) =
      if P(x) then x::filter(P, xs) else filter(P, xs);
```

（裏面に続く）

(5) 以下に示すのは2分木を表すデータ型である。

```
datatype 'a btree = Empty | Node of 'a * 'a btree * 'a btree;
```

これを用いて、2分木中の節点のラベルを帰りがけ順に整列したリストを返す関数 postorder: 'a btree -> 'a list を実装せよ。例えば postorder(Node(1, Node(2, Empty, Empty), Node(3, Empty, Node(4, Empty, Empty)))) の評価値は [2, 4, 3, 1] となる。（15点）

解答例

1. 型

(1)

a) (#"a", "foo", [[true, false, false], nil], false)

b) ([((1, 2), 3), ((4, 5), 6)], (3.5, ["foo", "bar", "baz"]))

(2) int list * int -> int list

2. MLプログラミング

(1)

```
fun abs(x) = if x >= 0.0 then x else ~x;
```

(2)

```
fun positive(nil) = nil
  | positive(x::xs) =
      if x > 0.0 then x::positive(xs) else positive(xs);
```

(3)

```
fun hd_tl(nil) = (nil, nil)
  | hd_tl(nil::ys) = hd_tl(ys)
  | hd_tl((x::xs)::ys) =
     let
       val (hds, tls) = hd_tl(ys)
     in
       (x::hds, xs::tls)
     end;
```

(4)

```
fun concat(L) = reduce(fn (x, y) => x ^ y, L);
```

(5)

```
fun postorder(Empty) = nil
  | postorder(Node(a, l, r)) = postorder(l) @ postorder(r) @ [a];
```