Tutorial 1 Pemrograman Deklaratif

Rabu, 25 Februari 2015

1. Tentukan tipe dari ekspresi-ekspresi di bawah ini (tulis di sebelah kanan). Jika tidak bisa, tulis "tidak bisa" dan jelaskan mengapa tidak bisa.

`7`	int
`2 + 5`	int (7)
`(2 + 5)`	int (7)
`"a" > "b"`	bool (false)
`3.0`	float (3.0)
`3.0 * 11.0`	float (33.0)
`2 * 2.0`	Tidak bisa karena 2 adalah int dan 2.0 adalah float
`(1, 5)`	int * int
`(9 - 4, 5)`	int * int (5, 5)
`(2.0, 3)`	float * int (2.0, 3)
`[(1,1); (2,2); (3,3)]`	(int * int) list
`([1;1], [2;2], [3;3])`	int list * int list * int list
`"1"`	string = "1"
`"2 + 5" + "1 + 1"`	string = "2 + 51 + 1"
`[["x";"y"];["z"]]`	string list list
`let abc = 123`	val abc : int = 123
`let jumlahkan1 x = x + 1`	int -> int
`let jumlahkan2 (x) = x + 2`	int -> int
`let jumlah (x, y) = x + y`	int * int -> int Kalau mau concat string, harusnya let jumlah (x:string, y) = x + y;;
`let jumlah x y = x + y`	int -> int -> int
`let fungsi z = 2015`	'a -> int z bisa diisi apa saja, int, string, float
`let f (x, y, z) = (x, y*z)`	'a * int * int ->'a * int
`let f (x, y, z) = [x; y*z]`	'a * int * int -> int list

`let rec f = function`
`| [] -> []`
`| x::xs -> x + 1::f xs`
	int list -> int list
	yg dilakukan adalah menjumlahkan setiap elemen list dengan 1

`let rec f = function`
`| [] -> 0`
`| x::xs -> x + (f xs);;`
	int list -> int
	Menjumlahkan semua isi elemen list

2. Buatlah fungsi-fungsi berikut.
   a. faktorial x
      faktorial x menerima input suatu bilangan bulat ≥ 1 dan mengembalikan hasil faktorialnya dengan syarat 1! = 1.

```
let rec faktorial x = if x = 1 then 1
                      else x * faktorial (x - 1);;
```


b. adaDuplikat x

adaDuplikat x menerima input suatu list dan memeriksa apakah ada pengulangan elemen dalam list. Jika ada, kembalikan true; jika tidak ada, kembalikan false.

```
let rec adaDuplikat = function
| [] -> false
| x::xs ->
    let rec isMember = function
    | (x, []) -> false
    | (x, y::l) -> (x = y) || isMember (x,l)
    if isMember(x, xs) then true else adaDuplikat(xs);;
```

c. nolGanjilSatuGenap x

fungsi ini menerima input suatu list berisi bilangan bulat dan mengembalikan suatu list bilangan bulat 0 atau 1. Jika bilangan pada list input adalah ganjil, diganti dengan 0. Jika genap, diganti dengan 1.

```
let rec nolGanjilSatuGenap = function
| [] -> []
| x::xs -> if x%2 = 1 then 0::nolGanjilSatuGenap xs
           else 1::nolGanjilSatuGenap xs;;
```

d. reverse x

Fungsi reverse x menukar urutan isi list dari belakang ke depan.

```
let rec rev = function
| ([], ys) -> ys
| (x::xs, ys) -> rev (xs, x::ys);;
```

e. palindrome x

Fungsi ini memeriksa apakah x adalah list yang palindrome atau bukan. Jika palindrome, kembalikan true; jika tidak, false. Contoh palindrome adalah [“a”; “b”; “a”], [1] atau [ ]

```
let rec palindrome = function
| [] -> true
| x::xs ->
      let rec reverse = function
      |[] -> []
      |(x::xs) -> reverse xs @ [x]

      let y = reverse (x::xs)
      if (x::xs)= y then true
      else false;;
```


3. Lakukanlah *tracing* untuk fungsi berikut. Apa yang dilakukan oleh fungsi ini? Tentukan jenis fungsi ini.

```
let rec fungsi = function
| (_, []) -> []
| ([], _) -> []
| (x::xs, y::ys) -> if x = y then true::fungsi(xs, ys)
                    else false::fungsi (xs, ys);;

fungsi([1;2;5;1;0], [1;3;3;1])
```

Jawab:

fungsi([1; 2; 5; 1; 0], [1; 3; 3; 1]) = true :: fungsi([2;5;1;0], [3;3;1])
= true :: (false :: fungsi ([5; 1; 0], [3; 1]))
= true :: (false :: (false :: fungsi ([1; 0], [1])))
= true :: (false :: (false :: (true :: fungsi ([0], [ ]))))
= true :: (false :: (false :: (true :: [])))
= true :: (false :: (false :: [true]))
= true :: (false :: [false; true])
= true :: [false; false; true]
= [true; false; false; true]

Jenis fungsinya adalah **'a list * 'a list -> bool list when 'a: equality**