プログラミング言語基礎論第 1 回レポート

Kwame Ackah Bohulu, 1631133

22-06-2017

[1]
```
primes :: Int ->[Int]
primes n =seive [2..]
where
seive (p:xs)= p:seive (take n [x|x <- xs, mod x p > 0])
seive[]=[]
```

[2]
```
poly :: (Num a) => [a] -> a -> a
poly [] v=0
poly (x:xs) v = x*v + poly xs v
```

[3]
```
fib2 :: Int -> Int
fib2 n = fibsub n (1, 1)
fibsub :: Int -> (Int, Int) -> Int
fibsub n (x,y) | n== 0 =x
| otherwise = fibsub (n-1) (y,x+y)
```

fib2 関数は 0.00 秒で値を計算して、メモリ量は 102984 bytes なので、fib2 関数は効率的である。

[5] (1).
```
data BETree a = BLeaf a | BNode a (BETree a) (BETree a)
deriving (Eq, Show)
```

(2).
```
sumBETree :: (Num a) => a -> BETree a -> a
sumBETree u (BLeaf x) = u + x
sumBETree u (BNode x lt rt) = sumBETree (x + sumBETree u lt) rt
```

```
depthBETree :: BETree a -> Int
depthBETree (BLeaf x) = 0
depthBETree (BNode x lt rt) = 1 + max (depthBETree lt) (depthBETree rt)
```

```
upAccBETree :: (Num a) => a -> BETree a -> BETree a
upAccBETree u (BLeaf x) = BLeaf (u+x)
upAccBETree u (BNode x lt rt) = BNode(sumBETree (x + sumBETree u lt)
rt) ((upAccBETree u lt)) (upAccBETree u rt)
```

[6]

```
f :: Int -> [Int]
f a = if a <= 0 then [0] else a : f (a-1)

    g :: (Num a) => ([a],[a]) -> [a]
g ([],ys)=[]
g (xs,[])=[]
g ((x:xs), (y:ys)) = (x+y) : g(xs,ys)
```