Funcții recursive Laborator 2

Scrieți următoarele funcții:

1. funcții peste Bool: și logic, sau logic, negație, nand, nor, implicație, dublă implicație;

```
and :: Bool -> Bool -> Bool
and False _ = False
and _ False = False
and _ _ = True
```

2. test de primalitate: isPrime :: Integer -> Bool;

Folosiți o funcție auxiliară hasDivisors astfel încât hasDivisors n a b testeaza daca numarul natural n are divizori intre numerele natural a si b.

- 3. implementați diferiți algoritmi pentru calculul cmmdc (algoritmul lui Euclid prin scăderi/împărțiri repetate, algoritmul binar).
- 4. este posibilă aplicarea unei optimizări pentru aducerea apelurilor recursive în poziție de coadă?
- 5. implementați algoritmi pentru calculul celui de-al n-lea număr Fibonacci.

```
fibo :: Integer -> Integer
fibo ... = ...
...
```

Implementați varianta cu acumulatori:

```
fiboaux :: Integer -> Integer -> Integer
fiboaux 0 a _ = a
fiboaux n a b = fiboaux ... ...
-- a si b sunt doua numere Fibonacci consecutive
fibo' :: Integer -> Integer
fibo' n = fiboaux n 0 1
```

Implementați varianta care funcționează în timp O(log(n)).

Folosiți un raționament ecuațional pentru a arăta că fibo și fibo' sunt echivalente funcțional.

- 6. implementați algoritmul extins al lui Euclid.
- 7. implementați funcția succ :: Integer -> Integer.
- 8. implementați recursiv (chiar dacă sunt ineficiente) următoarele funcții: adunarea a două numere naturale, folosind succ; înmulțirea a două numere naturale, folosind adunarea; ridicarea la putere, folosind înmulțirea.
- 9. implementați funcțiile mod și div pentru numere naturale.