Programowanie funkcyjne – laboratorium 3
21.10.2009 r.

Dana jest następująca definicja typu drzew binarnych z wartościami typu `int` w wierzchołkach wewnętrznych.

```
type int_tree = Leaf | Node of int_tree * int * int_tree;;
```

1. Napisz funkcję `full`, która wywołana z dwoma argumentami `full n d` utworzy pełne drzewo o wysokości `d` z liczbą `n` w każdym wierzchołku. Drzewo o wysokości `0` to liść.

2. Napisz funkcję `size`, która oblicza liczbę wierzchołków wewnętrznych danego drzewa. Powinno być: `size (full n d) = 2^d - 1`, dla $d \geqslant 0$.

   ```
   size Leaf = 0
   size (Node (Node (Leaf, 1, Leaf), 0, Leaf)) = 2
   ```

3. Napisz funkcję `sum`, która zwraca sumę liczb w drzewie. Powinno być:
   `sum (full n d) = n * size (full n d) = n * (2^d - 1)` .

4. Napisz funkcję `depth`, która zwraca największą głębokość liścia w danym drzewie. Powinno być: `depth (full n d) = d`, dla $d \geqslant 0$.

5. Odległością między dwoma wierzchołkami w drzewie nazywamy minimalną liczbę krawędzi jakie trzeba przejść z jednego wierzchołka do drugiego. Średnicą drzewa nazwiemy maksymalną odległość między dwoma węzłami w drzwie. Przyjmujemy, że średnica pustego drzewa jest równa 0. Napisz procedurę `srednica: tree -> int`, która oblicza średnicę danego drzewa.

Drzewa BST

1. Napisz funkcję `bst_insert`, która wstawia liczbę do drzewa, zachowując własność BST.

2. Napisz funkcję `bst_insert_list`, która tworzy drzewo BST z listy liczb, wstawiając je kolejno do pustego drzewa.

3. Napisz funkcję `elem_list`, która mając dane drzewo BST tworzy listę jego wierzchołków odwiedzając je w porządku infiksowym, tzn. najpierw odwiedza lewe poddrzewo, potem korzeń, a następnie prawe poddrzewo.

4. Zauważ, że lista zwracana przez `elem_list` dla dowolnego drzewa BST jest posortowana. Wykorzystaj tę własność do napisania funkcji `bst_sort`, która bierze listę liczb i zwraca ją posortowaną.

5. Zaimplementuj takie operacje na zbiorach (reprezentowanych jako drzewa BST) jak suma, przecięcie i różnica.

Inne struktury danych

1. Zdefiniuj typ reprezentujący drzewa o wierzchołkach dowolnego skończonego stopnia. Zdefiniuj kilka procedur operujących na takich drzewach, np. procedury wyznaczające liczby w porządku prefiksowym i postfiksowym.

2. Dany jest typ rezprezentujący (mało efektywnie) liczby naturalne:

   ```
   type nat = ZERO | SUCC of nat;;
   ```

   Zaimplementuj operacje arytmetyczne na takich liczbach.