Lista 4

Zadanie 1 (2p.)

Napisz funkcję rozstrzygającą czy dana lista jest palindromem, ale taką, która wykonuje co najwyżej [n/2] wywołań rekurencyjnych, gdzie n jest nieznaną długością listy (nie potrzeba i nie wolno jej liczyć).

Zadanie 2 (2p.)

Rozważmy typ danych dla drzew binarnych, zdefiniowany następująco:

```
type 'a btree = Leaf | Node of 'a btree * 'a * 'a btree
```

Mówimy, że drzewo jest zbalansowane, jeśli dla każdego węzła v liczby węzłów w lewym i prawym poddrzewie zakorzenionym w v różnią się co najwyżej o 1.

1. Napisz efektywną funkcję sprawdzającą czy dane drzewo jest zbalansowane.
2. Napisz funkcję, która dla zadanej listy etykiet tworzy zbalansowane drzewo, dla którego tę listę można otrzymać przechodząc je w porządku preorder.

Zadanie 3 (2p.)

Napisz możliwie efektywne funkcje przejścia wszerz oraz przejścia w głąb (preorder) dla obu reprezentacji drzew wielokierunkowych podanych na wykładzie (zadanie 5 z listy kontrolnej do wykładu 4).

Zadanie 4 (6p.)

Zdefiniuj typ służący do reprezentacji formuł rachunku zdań składających się ze zmiennych zdaniowych, negacji, koniunkcji i alternatywy.

1. Napisz funkcję sprawdzającą, czy dana formuła jest tautologią. W tym celu należy generować kolejne wartościowania zmiennych zdaniowych występujących w danej formule i sprawdzać dla nich wartość formuły. W przypadku, gdy formuła nie jest tautologią, funkcja powinna, oprócz tej informacji, podać jedno z wartościowań, dla których formuła nie jest prawdziwa.
2. Napisz funkcję, która przekształca zadaną formułę w formułę jej równoważną w negacyjnej postaci normalnej, tj. w której negacja występuje tylko przy zmiennych zdaniowych.
3. Napisz funkcję, która przekształca zadaną formułę w formułę jej równoważną w koniunkcyjnej postaci normalnej.
4. Napisz funkcję sprawdzającą (syntaktycznie), czy zadana formuła jest tautologią, korzystając z faktu, że każdą formułę można przedstawić w koniunkcyjnej postaci

normalnej.

5. Napisz funkcję, która przekształca zadaną formułę w formułę jej równoważną w dysjunkcyjnej postaci normalnej.
6. Napisz funkcję sprawdzającą (syntaktycznie), czy zadana formuła jest sprzeczna, korzystając z faktu, że każdą formułę można przedstawić w dyzjunkcyjnej postaci normalnej.

Zadanie 5 (2p.)

Funkcja jest napisana w stylu przekazywania kontynuacji (continuation-passing style, CPS), jeżeli przyjmuje dodatkowy argument funkcyjny zwany kontynuacją, który reprezentuje całą resztę obliczeń jakie mają zostać przeprowadzone po powrocie z tej funkcji. W konsekwencji, funkcje w CPS-ie zwracają wynik wywołując swoją kontynuację, a wszystkie wywołania w programie w CPS-ie są ogonowe. Na przykład, funkcja licząca silnię napisana w CPS-ie wygląda następująco:

```
let rec fact_cps n k =
  if n = 0
  then k 1
  else fact_cps (n-1) (fun v -> k (n*v))
```

Kontynuacja początkowa przekazana funkcji `fact_cps` mówi co zrobić z wynikiem obliczenia silni zadanej liczby. Typowe wywołanie funkcji `fact_cps` dostaje identyczność jako kontynuację początkową (gdy silnia jest obliczona, wystarczy ją zwrócić):

```
let fact n =
  fact_cps n (fun v -> v)
```

Dla drzew binarnych z zadania 2, napisz funkcję `prod : int btree -> int`, która liczy iloczyn wszystkich wartości w drzewie. Zapisz tę funkcję w CPS-ie, a następnie zmodyfikuj otrzymaną funkcję tak by w przypadku napotkania wartości 0 funkcja wykonała bezpośredni skok do miejsca swego wywołania, bez krokowego powracania z rekursji.