NPRG005 Test základů neprocedurálního programování

_____

jméno a příjmení

Pište prosím čitelně, nečitelný text bude považován za chybný.
Plné bodové ohodnocení se poskytuje za stručné, elegantní a korektní řešení.
Body se odečítají za řešení zbytečné komplikované či nestrukturované.
U všech úloh můžete předpokládat, že vstup je zadán korektně.
*Můžete používat standardní predikáty (např. **member**/3, **append**/3, **reverse**/2,...) jazyka Prolog či standardní funkce (viz přiložený seznam) jazyka Haskell, které byly na přednášce. V Prologu prosím nepoužívejte predikáty typu **assert** a **retract** či **bagof**, **setof** a **findall**.*
Je třeba získat alespoň 5 bodů z úloh 1-2, alespoň 5 bodů z úloh 3-4, a alespoň 12 bodů celkem.
Čas: 1h 20min

1. (5 bodů) **Prolog.** Profesor Hammerstein definoval predikat **setrid/2** takto:

```
% setrid(+X,-Y) :- Y je seznam přirozených čísel X setříděný vzestupně
setrid(X,Y) :- append(A,[H1,H2|B],X), H1 > H2, !,
               append(A,[H2,H1|B],X1), setrid(X1,Y).
```

zapomněl však na klauzuli, která definuje bázi rekurze.

(a) Doplňte jednu chybějící klauzuli (před nebo za uvedené pravidlo).

```
setrid(X,Y) :- append(A,[H1,H2|B],X), H1 > H2, !,
               append(A,[H2,H1|B],X1), setrid(X1,Y).
```

(b) V definici je použit řez (**!**). Jde o *zelený* (nemění deklarativní význam) či *červený* řez (mění d.v.) ? Vysvětlete!

(c) Jaký známý třídící algoritmus výše uvedený kód implementuje? Pokud neznáte název, můžete alespoň slovně popsat, jak **setrid/2** funguje.

2. (5 bodů) **Prolog.** Binární strom, jehož vrcholy jsou ohodnoceny (ne nutně různými) přirozenými čísly, je reprezentován
   - termem `t(V,L,P)`, kde `V` je ohodnocení kořene, zatímco `L` a `P` levý a pravý podstrom
   - atomem `nil`, jde-li o prázdný strom.

   Definujte predikát `trans/2`, který k zadanému vstupnímu stromu `T` sestrojí strom téhož tvaru jako `T`, v němž bylo ohodnocení každého vrcholu nahrazeno maximálním ohodnocením, které se v `T` vyskytuje.

   Důležité: Plné bodové ohodnocení bude uděleno jen řešení, které
   - použije pouze jediný (rekurzivní) průchod stromem `T`
   - nebude obsahovat více než 4 klauzule.

3. (5 bodů) **Haskell.** Definujte funkce

`prevod1` `cislo puvodni`

pro převod čísla z z číselné soustavy o základu `puvodni` do dekadické číselné soustavy, a

`prevod2` `cislo nova`

pro převod čísla z dekadické do číselné soustavy o základu `nova`.

Příklad:

```
> prevod1 [1,1,1,0] 2  -- převede binární 1011 do desítkové soustavy
14
> prevod 33 16  -- převede dekadické 33 do hexadecimální soustavy
[2,1]
```

(a) Doplňte typové signatury definovaných funkcí

```
prevod1 ::
prevod2 ::
```

(b) Definujte funkci **`prevod1`** s využitím rekurze.

(c) Sestavte alternativní definici funkce **`prevod1`** s využitím alespoň jedné z funkcí `map`, `filter`, `foldr` či `foldl`, ale bez použití rekurze.

(d) Definujte funkci **`prevod2`** s využitím funkce **`unfold`** definované následovně:

```
unfold :: (t -> Bool) -> (t -> (a, t)) -> t -> [a]
unfold done step x =  if done x then []
                                else let (y,ys) = step x
                                     in y: unfold done step ys
```

4. (5 bodů) **Haskell.** Dva (kořenové) stromy s ohodnocenými vrcholy jsou *izomorfní*, pokud lze jeden obdržet z druhého vhodnou permutací podstromů pro každý vrchol. Různé vrcholy mohou být ohodnoceny stejnými hodnotami.

(a) Definujte polymorfní datový typ **Strom a** pro reprezentaci takového obecného (nikoliv pouze binárního) stromu s vrcholy ohodnocenými hodnotami typu **a**.

(b) Nad touto reprezentací definujte funkci

isom :: Strom a -> Strom a -> Bool

která zjistí, zdali jsou dva zadané stromy izomorfní.