1. Zkouška 2. 6. 2015 (Dvořák, Hric)

Prolog:

1) Definujte predikát tranverse(+Strom,-OhodnocenýStrom), který zkopíruje strukturu stromu Strom do OhodnocenýStrom s tím, že ke každému vrcholu přidá číslo N, které znamená pořadí v preOrder průchodu a číslo M, které znamená pořadí v postOrder průchodu. Ideálně jedním průchodem stromem.

```
tranverse(t(t(nil,l,nil),v,t(nil,p,nil)),X).
X = t(t(nil,l-2-1,nil),v-1-3,t(nil,p-3-2,nil))
```

2) Definujte predikát atmost2(+seznam,-seznam2), který dostane seznam ohodnocených binárních proměnných a vrátí seznam všech možných ohodnocení stejných proměnných, které se od původního seznamu liší v maximálně 2 proměnných.

```
atmost2([x-true,y-false],X).
X = [[x-true,y-false],[x-false,y-false],[x-false,y-true],[x-true,y-
true]]
```

Haskell:

3) Napište funkci, která převede slovo "abbbccac" na seznam, kde jsou po sobě jdoucí stejné znaky sjednoceny do dvojice (znak, početVýskytů). K reprezentaci výsledku použijde Either viz příklad:

```
rle::Eq a=>[a]->[Either a (a,Int)]
rle [a,b,b,b,c,c,a,c] = [Left a, Right (b,3), Right (c,2), Left a, Left
c]
```

Napište, jakou definici bude mít inverzní funkce, která naopak převede kontrahovaný seznam na původní slovo a potom ji napište s pomocí concat a map.

4) Napište fold pro binární stromy. Napište funkci využívající váš fold, která pro daný binární strom spočítá počet jeho listů.

```
data BTree a = Nil | Vertex (BTree a) a (BTree a)
fold::b->(b->a->b->b)->BTree a->b
```

Velká úloha:

Hypergraf H = (V,E), kde V je množina vrcholů a E je libovolná podmnožina vrcholů (ne nutně dvouprvková). Lineární hypergraf je pak hypergraf, jehož žádné 2 hrany se neprotínají ve víc než jedno bodě. Erdösova hypotéza říká, že každý hypergraf se dá hranově dobře (stejně obarvené hrany mají prázdný průnik) obarvit pomocí nejvýše |V| barev. Napište predikát/funkci, který(á) generuje všechny lineární hypergrafy a otestujte pro každý, zda ho lze obarvit maximálně n barvami. U obecných hypergrafů je hledání takového obarvení NP-Úplný problém, ale pro lineární se předpokládá (nikdo Erdösovu hypotézu za 40 let nevyvrátil), že takové obarvení existuje. Použijte proto pro hledání toho obarvení nějakou heuristiku.

Já jsem jako heuristiku jednak setřídil hrany, které se chystám obarvit podle počtu vrcholů v nich obsažených sestupně a pak jsem ještě střídal použité barvy (místo abych jednou barvou barvil vše co můžu a měnil barvu až když nemůžu, tak jsem právě použitou barvu dával na konec seznamu pro barvení další hrany). Tyto heuristiky se mu líbily, neříkal, jestli doufal v lepší.

2. Zkouška 25. 5. 2014 (Dvořák, Hric)

První část:

1.

- napište predikát rotace/2 na rotování seznamu. Nesmíte použít žádné pomocné predikáty. (V lineárním čase) (pouze 3 verze)
- v konstantním čase, jakou potřebujete strukturu? Ukažte na [1,2,3]
- napište rotace/2 (pouze 2 verze) v konstantním čase

2.

- máte ČUM ([a>-b, a>-c, ...]). Vypište všechny neporovnatelné dvojice (ve formě seznamu), tzn například [b-c, ...]

3.

- Máte XML strom

```
data Tree a = T a [Tree a]
```

  - vypište ho

4.

- Máte Multiset, napsat instanci třídy Ord (m1 < m2 <==> existuje prvek v m2\m1, ktery je vetsi, nez vsechny prvky v m1\m2 - mnozinovy rozdil)

Druhá část:

Máte N truhel, N+1 klíčů, v každé truhle je právě jeden klíč, jeden máte na začátku.
Každý klíč má svou barvu a každá truhla má barvu, truhlu otevřete jen klíčem té samé barvy. Zároveň když použijete klíč, už ho nemůžete použít znovu (zůstane v zámku)?
Existuje posloupnost otvírání truhel tak, že otevřete všechny truhly? Pozn. máte to udělat polynomiálně

---

3. Zkouška 26.6.2014 (Dvořák + Hric)

Prolog

1. Byl dán seznam objektů, kde každý objekt obsahuje dvojice klíč-hodnota. Úkolem bylo pro každý klíč vypsat jakých hodnot nabývá.
Pokud nějaký objekt klíč neobsahuje, tak je hodnota undefined. Objekty jsou prostě seznamy.

```
[
 [ jmeno-"xy", vek-30, vaha-90],
 [ jmeno-"xyz", vek-35, vaha-80]
 [ jmeno-"ab",  vaha-80]
]
[ jmeno-["xy", "xyz", "ab"], vek-[30, 35, undefined], vaha-[90, 80,
80]]
```

2. Mame dany nedokonaly BVS, mame vypisat dvojice vrcholov, ktore porusuju podmienky BVS. Kazdy vrchol a list ma nejaku hodnotu, co je unikatne cele cislo. Uz to tu par krat bolo.
Datova struktura:

```
data BVS a = N nil | NT (BVS a) a (BVS a)
```

(alebo nieco podobne)

Haskell

3. HTML serializer. Mame dany obecny strom, ktory ma vo vrcholoch a listoch nazvy HTML tagov, napr. "html", "body", "a", atd. Cielom bolo:
a) napisat typ konstruktoru tej datovej struktury
b) vypisat tagy do stringu podla DFS priechodu stromom zlava - teda vlastne tak, aby to bolo validne HTML.
Signatura:

```
vypis (Num a, Ord a):: NTree a -> String
```

Priklad:

```
Strom:          html
               /    \
            head    body
                   /    \
                  a     h2

Vypis: <html><head></head><body><a></a><h2></h2></body></html>
```

4. Mame body v niekolkodimenzionalnom priestore. Vzdialenost medzi bodmi je urcena pomcou Manhattanskej metriky. Funkcia dostane zadanu dvojicu bodov (prihradok), a zoznam dalsich bodov. Singatura funkcie bola:

```
zarad :: (Num a, Ord a) => ([a], [a]) -> [[a]] -> ([[a]], [[a]])
```

Kazdy bod zo zoznamu bolo treba zaradit do tej blizsej z tych 2 prihradok, teda do tej, ktora je blizsie podla Manhattanskej metriky.
Napriklad:

```
> zarad ([0,0] [5,5]) [[1,2], [6,4], [0,-1]]
([[0,0], [1,2], [0,-1]], [[5,5], [6,4]])
```

Big One

Je dany pocet strojov n, a zoznam vyrobkov (kazdy vyrobok ma dane cislo, dobu kolko trva jeho vyrobenie, cas odkedy je mozne ho vyrobit, a cas dokedy musi jeho vyroba skoncit). Mozeme predpokladat, ze doba vyrobenia sa vojde do tohoto intervalu od-do (teda kazdy vyrobok je vyrobitelny). Stroje su navzajom zamenitelne, vyrobky tiez. Kazdy stroj moze samozrejme naraz vyrabat len jeden vyrobok.
Ulohou bolo vratit rozvrh vyroby (teda zoznam zaznamov typu [cislo vyrobku, cas zacatia vyroby, cislo stroja]), tak aby sa maximalizoval pocet vyrobenych vyrobkov. Nemusia sa dat vyrobit vsetky. Malo sa to riesit s heuristikou. Dvorak nam poradil, aby sme to pisali zhora, teda pomocne funkcie az nakoniec, pre pripad, ze by sme to nestihli.
Uloha bola velmi necakane NP uplna. Podla mna bol na to ovela lepsi Prolog, ale viem, ze to par ludi robilo aj v Haskelli. Ja som to robila tak, ze som sa najprv pokusila najst rozvrh, ktory vyrobi vsetky vyrobky (teda m vyrobkov), ak sa to nepodari, skusi vyrobit m-1... atd., az po skusi vyrobit jeden vyrobok. Ak sa mu nepodari ani jeden, zlyha. Rozvrh som sa snazila vyrobit pomocou predikatov between a kontrolovania podmienok.
Bolo treba splnit podmienky: jeden stroj nikdy nevyraba viac veci naraz, vyroba kazdeho vyrobku prebieha v jeho povolenom intervale

4. Zkouška 26.5.2014 (Dvořák + Hric)

Prolog:

1) Definujte predikát orez(+Strom,+D,+H,-VStrom), který ve Stromu ponechá pouze uzly V, že D <= V <= H. Už bylo min 2x.

2) Máme neorientovaný graf bez smyček, reprezentovaný pomocí seznamu sousedů. Napište predikát trojuhelniky(G,V), který ve V vrátí seznam všech trojúhelníků (třech vrcholů, které mezi sebou všechny mají hrany). Trojúhelníky by se v seznamu neměly opakovat.

Př:

```
troj([a->[b,c,d],b->[c,a],c->[b,d,a],d->[a,c],e->[]],V).
V = [troj(a,b,c),troj(a,c,d)]
```

Haskell:

1) Násobení řídkých polynomů -> mějme řídké polynomy reprezentované pomocí [(nenulový koeficient,exponent)]. Definujte pro ně datový typ (nezapomeňte na nulový polynom) a napište funkci mult (i její datovou signaturu), která bude řídké polynomy násobit.

Kdo neví (jako já jsem nevěděl) co je řídký polynom -> u spousty exponentů je nulový koeficient (exponenty prostě nejdou po 1, ale skáčou), ty samozřejmě nejsou v dané reprezentaci.

```
data Ridky a = Ridky [(Int,Int)] | Void
Postup: vynásobit všechno se všim, posčítat závorky se stejným
exponentem.
```

2) Máme zadaný datový typ (Bag a = Item a | Items [a]) a typ funkce fold, která má tuto strukturu procházet. Máme definovat tuto funkci a poté ještě funkci listy, která vrátí seznam všech položek v Item a v této datové struktuře.

```
data Bag a = Item a | Items [Bag a]

-- a) Definovat funkci fold pro obecny pruchod datovou strukturou.
fold :: (a -> b) -> ([b] -> b) -> Bag a -> b
fold func listFunc (Item x) = func x
fold func listFunc (Items xs) = listFunc [fold func listFunc bag | bag <-
xs]

-- b) Definovat funkci listy, ktera vrati seznam polozek s datovymi
konstruktory Item.
listy :: Bag a -> [a]
listy bag = fold makeList joinLists bag

makeList :: a -> [a]
makeList x = [x]

joinLists :: [[a]] -> [a]
joinLists [] = []
```

```
joinLists (x:xs) = x ++ joinLists xs

-- > listy (Items [Item 1, Item 2, Items [Item 3, Item 4], Item 5])
-- [1,2,3,4,5]
```

Big One:
Mějme x1..xn, y1..yn posloupnosti, P cenu vyškrtnutí, I a J maximální počet po sobě jdoucích vyškrtnutí (pro posloupnost x resp y). Představte si, že jsou posloupnosti pod sebou a nyní máte prvky z x spárovat s y přímkami tak, že se nikde nic nekříží. V posloupnostech můžete vyškrtávat prvky, abyste je nemuseli párovat. Pro spárované xi s yi je cena abs(xi - yi). Pro celé možné spárování je tedy cena součet cen párů + (počet vyškrtnutých znaků * P). Najděte spárování s nejmenší cenou.
Můj postup: vygeneruju si všechny možnosti vyškrtnutí pro x a y -> zazipuju [zip a b | a <- moznosti x [argumenty], y <- moznosti y [argumenty]]. Doporučuju si v generovaných možnostech v posledním prvku držet penalizaci za vyškrtávání. Pak zjistím nejnižší cenu takových spárování a vyhodím všechny (oni chtěli jen jedno), které tuto cenu má.

---

5. **Zkouška 21.5.13 (Dvořák+Hric)**

   Prolog (pouze std predikáty z přednášky, bez použití assert, findall a dalších užitečných)
   1) Rotace seznamu v konstantním časem (pomocí rozdílových seznamů)

   2) dostanete graf a vrchol - máte najít každou nezávislou množinu vrcholů obsahující zadaný vrchol co největší do inkluze (může jich být víc! pozor na definici "co do inkluze")
   Reprezentace grafu je následující graf(Vrcholy, Hrany), kde Vrcholy=[Int]; Hrany=[v->[u, w, z], x->[y,z]]

   Haskell (papír s fcemi dostanete, když použijete jiné, musíte je na ústním naprogramovat)
   3) Reprezentujte dlouhá a neomezeně přesná čísla seznamem číslic a pozicí desetiné čárky; naimplementujte násobení těchto čísel

   4) zipWith na obecných stromech (hodnoty ve všech listech) - dostanete funkci, dva stromy a defaultní hodnotu a máte vrátit strom. Je potřeba rekurzivně se volat na syny a když v jednom stromu syn chybí a ve druhém je (nebo nopak) do prvního si "přimyslet" nový vrchol s defaultní hodnotou

   1 velká úloha
   Zobecnění hamiltonovské kružnice
   Dostanete (ohodnocený, orientovaný) graf a jeho 2 disjunktní podmnožiny vrcholů A, B takové, že |A|=|B|=N. Vašim úkolem je vypsat N (hranově i vrcholově) disjunktních cest vedoucích z A do B (tedy z každého vrcholu v A povede jedna; do každého vrcholu v B povede jedna). Navíc cesty musí projít projít všemi vrcholy. Navíc pokud je možných řešení jak cesty vést více, vyberte to, jehož hrany mají nejmenší součet délek. Pokud je i tak řešení více, vyberte takové, které má nejdelší cestu nejkratší možnou. Navíc se pokuste o vhodnou heuristiku (problém je *překvapivě* NP-úplný). Heuristika nebyla na plný počet vyžadována, ale pokud jste prohledávali do šířky, vyplatilo se jít nejkratší možnou cestou a potom další výpočty chytře ořezávat už na začátku, ale jak říkám - nebylo to podstatné.

6. **19.06.2015 - Dvořák, Hric**

Malé příklady

1. **Prolog** Napište predikát splay(+Hodnota, +Strom, -Vystup) provede operaci splay nad binárním vyhledávacím stromem Strom, která pomocí rotací přesune uzel s hodnotou Hodnota (pokud ve stromu není tak jejího přímého předchůdce či následníka) do kořene stromu a vrátí nový strom ve Vystup.
2. **Prolog** Napište predikát zlepsirez(+Graf, +Vrcholy1, +Vrcholy2, -OutV), který pro zadaný ohodnocený neorientovaný graf Graf a řez (definovaný pomocí dvou disjunktních množin vrcholů Vrcholy1 a Vrcholy2) najde vrchol, který když přesuneme do opačné skupiny vrcholů řezu, tak dostaneme řez s lepší cenou. Graf byl zadaný šikovně a bylo možné si ho případně upravit aby vyhovoval více.
3. **Haskell** Máme zadanou matici (jako list listů), napište funkci, která pro danou matici vrátí všechny dvojice indexů (x,y) takových, že podmatice (1,1) (x,y) je kladná (každý prvek je kladný) a zároveň x i y je největší možné. Výstupem mají být všechny takovéto dvojce

   $$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & -8 & 9 \end{pmatrix}$$

   První řádek (chápáno jako matice) nesplňuje zadání, protože rozšířením na první dva řádky dostaneme větší matici, která je pořád kladná. Naopak první sloupec jej splňuje, protože y-ově rozšířit nejde (OutOfBoundsException) a x-ově taky ne, protože (kvůli -8) by nebyla matice kladná. [(1,3), (3,2)] je řešení v tomto případě.
4. **Haskell** Je dán strom (ne nutně binární) a máme očíslovat jeho vrcholy v pre-order pořadí (viz. prologovský příklad z kázkové písemky na webu předmětu), Strom je data NT a = N a [NT a]

Velký příklad

Máte orientovaný graf a ke každé hraně máte seznam dvojic reálních čísel. Cílem je nějak ohodnotit vrcholy libovolnými čísly a hrany jednou dvojící ze seznamu tak, aby odchylka byla co nejmenší. Odchylka je součet odchylek všech hran.

Odchylka hrany s hodnotou počátečního vrcholu v_1, hodnotou koncového vrcholu v_2 a ohodnocením hrany (e_1, e_2) je $|v_1 - e_1| + |v_2 - e_2|$. Máte zvolit vhodnou heuristiku.