Vlastnosti struktry HEAP

Skupina: **10**
Řešitelé: **Stanke Michal**, **Timr Marek**, **Voříšek Lukáš**

Zadání úlohy

Specifikujte vlastnosti datové struktury "heap" s maximálním kořenem. Ukažte, že specifikace generuje pouze heap struktury do zadaného počtu uzlu (např. vlastnosti stromu, vlastnosti velikosti klíču, apod.)

Kód řešení

```
module Heap
open util/ordering[Level] as LO
open util/ordering[Node] as NO

// Signatura uzlu, obsahuje svou hloubku (level), rodiče (p), potomky (l,r) a
hodnotu (v).
sig Node {
        level: one Level,
        p: lone Node,
        l: lone Node,
        r: lone Node,
        v: one Int
}

// Signatura hloubky uzlu.
sig Level {}

// Halda má pouze jeden kořen.
// = existuje jen jeden uzel, který nemá rodiče
// = všechny uzly, které nemají rodiče, jsou v pořadí uzlů první
// = všechny uzly, které nemají rodiče, jsou nejvýše (v kořeni stromu)
fact JedenRoot {
        one n : Node | no n.p
        all n : Node | no n.p => n = NO/first
        all n : Node | no n.p => n.level = LO/first
}

// Rodič je vždy ve stromu výše než jeho potomek.
fact RodicMaMensiLevel {
        all n : Node | one n.p => n.level = LO/next[n.p.level]
}

// Potomek má odkaz na správeného rodiče.
// = kazdy uzel (n) ma potomka (p i l) takoveho, ze jeho rodicem je opet tento
uzel (n)
// = pravy (p) a levy (l) potomek uzlu (n) nejsou shodne
fact DiteJeVRodici {
        all n : Node | one n.l => n.l.p = n
        all n : Node | one n.r => n.r.p = n
        all n : Node | one n.r && one n.l => n.r != n.l
}
```

```
// Pokud má uzel pravého potomka, mé i levého (halda se plní zleva).
fact NejprveLeve {
        all n : Node | one n.r => one n.l
}

// Ke každému uzlu (n), který má rodiče, existuje uzel rodiče, pro kterého je
levým nebo pravým potomkem.
// = průnik (n) s levým a pravým potomkem jeho rodiče je jeden (on sám)
fact JePotomek {
        all n : Node | one n.p => one (n & (n.p.l + n.p.r))
}

// Žádný uzel není svým vlastním rodičem ani potomkem.
// = není uzel takový, aby byl obsažen v množině svého rodiče, jeho rodiče atd.
// = není uzel takový, aby byl obsažen v množině potomků svých potomků, jejich
potomků atd.
fact NeniVlastniRodicAniPotomek {
        no n : Node | n in n.^p
        no n : Node | n in (n.l.^(r + l) + n.r.^(r + l))

}

// Halda je odshora plná.
// = pro všechny dvojice uzlů (m,n) platí, že je-li m o úroveň hlouběji než m, má
m pravého i levého potomka
fact PlnyPredchoziLevel {
        all n : Node | all m : Node | one n.l && n.level = LO/next[m.level] => #
(m.r + m.l) = 2
}

// Spodní patro haldy se plní zleva.
// = pro všechny uzly (n) platí, že mají-li oba potomky, pak pravý nasleduje po
levém
// = pro všechny uzly (n) platí, že mají-li predchozí uzel a levého potomka, pak
ten následuje za nimi
fact PlniSeZLeva {
        all n : Node | one n.l && one n.r => n.l.next = n.r
        all n : Node | one NO/prev[n] && one n.l => NO/prev[n].r.next = n.l
}

// Žádne dva různe uzly nemají stejnou hodnotu.
fact JineHodnoty {
        all n : Node | all m : Node | n != m => n.v != m.v
}

// Všechny hodnoty jsou kladné.
fact VsechnyHodnotyKladne {
        all n : Node | n.v >= 0
}

// Hodnota v každém uzlu je menší než hodnota jeho rodiče.
fact VlastnostMaxHepy {
        all n : Node | one n.p => n.p.v >= n.v
}

// Assert: Existuje nanejvýš jeden uzel, který má jenom levého potomka a nemá
pravého.
pred jedinacek[] {
        lone n: Node | one n.l && no n.r
}
// Assert: Existuje jenom jeden uzel, který nemá rodiče.
pred oneRoot[] {
        one d: Node | no d.p
```

```
}
// Assert: Neexistuje žádný uzel, jehož hodnota by byla vyšší než hodnota v jeho
rodiči, pokud jej má.
pred PotomekMensiNezRodic[] {
         no n : Node | one n.p && n.v > n.p.v
}

check {jedinacek and oneRoot and PotomekMensiNezRodic} for 12 Node, 5 Level,8 Int
```