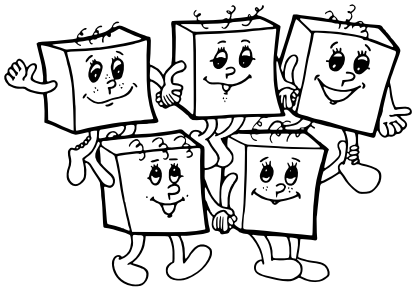
OLYMPIÁDA V INFORMATIKE

NA STREDNÝCH ŠKOLÁCH

http://oi.sk/



tridsiaty tretí ročník

školský rok 2017/2018

zadania celoštátneho kola, deň 1
kategória A

Priebeh celoštátneho kola

Celoštátne kolo 33. ročníka Olympiády v informatike, kategórie A, sa koná v dňoch 21.-24. marca 2018. Na riešenie úloh prvého, teoretického dňa majú súťažiaci 4,5 hodiny čistého času. Rôzne úlohy riešia súťažiaci na samostatné listy papiera. Akékoľvek pomôcky okrem písacích potrieb (napr. knihy, výpisy programov, kalkulačky) sú zakázané.

Čo má obsahovať riešenie úlohy?

- Slovne popíšte algoritmus.
  Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného algoritmu/programu.
- Zdôvodnite správnosť vášho algoritmu.
- Uveďte a zdôvodnite jeho časovú a pamäťovú zložitosť.
- Podrobne uveďte dôležité časti algoritmu, ideálne vo forme programu v Pascale alebo C/C++.
- V prípade, že používate vo svojom programovacom jazyku knižnice, ktoré obsahujú implementované dátové štruktúry a algoritmy (napr. STL pre C++), v popise algoritmu stručne vysvetlite, ako by ste napísali program s rovnakou časovou zložitosťou bez použitia knižnice.

Hodnotenie riešení prvého (teoretického) dňa

Za každú úlohu môžete získať od 0 do 10 bodov.
Pokiaľ nie je v zadaní povedané ináč, najdôležitejšie dve kritériá hodnotenia sú v prvom rade **správnosť** a v druhom rade **efektívnosť** navrhnutého algoritmu. Na výslednom počte bodov sa môže prejaviť aj kvalita popisu riešenia a zdôvodnenie tvrdení o jeho správnosti a efektívnosti.
Efektívnosť algoritmu posudzujeme vypočítaním jeho časovej zložitosti – funkcie, ktorá hovorí, ako dlho vykonanie algoritmu trvá v závislosti od veľkosti vstupných parametrov. Nezávisí pri tom na konštantných faktoroch, len na rádovej rýchlosti rastu tejto funkcie.
V zadaní úlohy môžu byť uvedené limity na veľkosť premenných. Tieto môžete použiť na odhad toho, ako dobré vaše riešenie je. Na počítači, ktorý vykoná miliardu inštrukcií za sekundu, vyrieši vzorové riešenie ľubovoľný povolený vstup nanajvýš za niekoľko sekúnd.

A-III-1 Kapitánka hľadá posádku

Bolo raz jedno dievčatko plné snov. Až jedného dňa vyrástlo, rozhodlo sa, že si jeden aj splní a že sa stane pirátskou kapitánkou. Keď si už zohnala klobúk, pásku cez oko a loď, chýbalo jej už len jediné: spoľahlivá posádka. A tak najprv zakotvila v prístave Port Royal, kde chce nejakú posádku naverbovať.
Keď sa rozkríklo, že sa verbuje nová posádka, zišiel sa na móle rad mladých nádejných pirátov, ktorí sa všetci chceli prihlásiť do kapitánkinej posádky. Kapitánka si ich očíslovala od 1 po n v poradí, v ktorom stoja v rade, a o každom z nich si poznačila číslo p_i hovoriace, aký je to dobrý pirát (čím väčšia hodnota, tým lepší).

Kapitánka vie, že pri výbere posádky sú dôležité dve veci: musia to byť dostatočne dobrí piráti a musia zároveň byť dobrá partia. Naverbovať dobrú partiu je ľahké: piráti, ktorí spolu dobre vychádzajú, určite stoja v rade vedľa seba, takže stačí, keď ako posádku zoberie nejaký neprázdny súvislý úsek uchádzačov.
Dobrá partia vám ale námornú bitku nevyhrá a naverbovať dostatočne schopnú posádku vôbec nie je jednoduché. Dôležitá je pri tom tzv. Denisova hranica. Denis je známym archetypom len-tak-tak použiteľného piráta. Nič mu nejde nejak hviezdne, ale všetky podstatné pirátske úlohy nakoniec nejak zvládne.

Pri vyhodnotení toho, či je posádka schopná plavby, je dôležitý koncept tzv. *stredného piráta*: ak by sa všetci piráti usporiadali podľa ich schopností, je to ten, ktorý by stál uprostred. Ak je pirátov párny počet, stredným pirátom je **lepší** z tých dvoch ktorí sú uprostred poradia.
Posádka je plavbyschopná práve vtedy, keď je stredný pirát v nej aspoň takým dobrým pirátom ako je Denis.

Súťažná úloha

V rade stojí n pirátov, očíslovaných od 1 po n v poradí, v ktorom stoja. Poznáme čísla $p_1, \ldots, p_n$ ktoré o každom z nich hovoria, aký je dobrý pirát. Tiež poznáme číslo d hovoriace to isté o Denisovi. (Samotný Denis nepatrí medzi uchádzačov a nedá sa ho naverbovať do posádky.)
Kapitánka si chce vybrať **neprázdny súvislý úsek** uchádzačov, a to tak, aby dostala plavbyschopnú posádku. Inými slovami, stredný pirát vo vybranom úseku musí byť aspoň tak dobrým pirátom ako Denis.
Zistite, koľkými spôsobmi si môže Kapitánka vybrať svoju posádku.

Formát vstupu a výstupu

V prvom riadku vstupu sú celé čísla n a d. V druhom riadku sú celé čísla $p_1, \ldots, p_n$.
Na výstup vypíšte počet spôsobov ako vybrať plavbyschopnú posádku ktorá bude dobrou partiou.

Obmedzenia a hodnotenie

Môžete predpokladať, že d aj všetky p_i sa zmestia do bežných celočíselných premenných.

Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 500$, môžete získať nanajvýš 3 body.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 5000$, môžete získať nanajvýš 5 bodov.
Za riešenie, ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 100\,000$, môžete získať 8 až 10 bodov podľa jeho presnej asymptotickej časovej zložitosti.

Príklad

vstup	výstup
4 30 10 40 20 30	7

Kapitánka má nasledovné možnosti: zobrať ako posádku len piráta 2, len piráta 4, hociktorých dvoch po sebe idúcich pirátov, pirátov 2+3+4, alebo všetkých štyroch pirátov.

Všimnite si, že ak by zobrala pirátov 2+3+4, stredným pirátom v posádke by nebol pirát 3 (ktorý teraz stojí v strede vybraného úseku) ale bol by ním pirát 4. Pirát 4 je taktiež stredným pirátom v prípade, že Kapitánka ako posádku zoberie všetkých štyroch uchádzačov.

A-III-2 Závažia

Jarka našla na povale sadu n exotických závaží. Jednotlivé závažia mali hmotnosti $m_1, \ldots, m_n$. Potešila sa, že si pomocou nich bude môcť vážiť všetko, čo len bude potrebovať. Samozrejme, namiesto jedného závažia vie Jarka použiť aj ľubovoľnú ich podmnožinu a odvážiť tak aj iné hmotnosti.

Súťažná úloha

Dokopy existuje 2^n podmnožín závaží. Predstavme si, že sme ich všetky zoradili podľa celkovej hmotnosti a očíslovali, začínajúc od jednotky. Ak má viacero podmnožín tú istú hmotnosť, môžeme ich zoradiť ľubovoľne. Pre dané n, jednotlivé hmotnosti závaží a dané relatívne malé číslo k nájdite hmotnosť množiny s číslom k.

Formát vstupu a výstupu

V prvom riadku vstupu sú kladné celé čísla n a k. V druhom riadku vstupu sú **kladné** celé čísla $m_1, \ldots, m_n$. Na výstup vypíšte hmotnosť podmnožiny, ktorá vo vyššie definovanom číslovaní dostane číslo k.

Obmedzenia a hodnotenie

O číslach m_i predpokladajte, že sú kladné a ich súčet sa zmestí do bežnej celočíselnej premennej. Tiež môžete predpokladať, že hmotnosti sú usporiadané, teda že platí $m_1 \leq m_2 \leq \cdots \leq m_n$.

Za riešenie ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 20$ môžete získať 2 body.
Za riešenie ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 500$ a $k \leq 500$ môžete získať 5 bodov.
Za riešenie ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 5000$ a $k \leq 5000$ môžete získať 7 bodov.
Za riešenie ktoré efektívne vyrieši ľubovoľný vstup s $n \leq 100\,000$ a $k \leq 100\,000$ môžete získať 10 bodov.

Príklady

vstup	výstup
3 5 1 1 2	2

Ak si závažia označíme A, B a C, tak máme osem množín závaží:

$$\emptyset,\ \{A\},\ \{B\},\ \{C\},\ \{A,B\},\ \{A,C\},\ \{B,C\},\ \{A,B,C\}$$

Ich hmotnosti sú 0, 1, 1, 2, 2, 3, 3, 4, čiže vyššie uvedené poradie je už usporiadané podľa hmotnosti. Poradové číslo $k = 5$ teda zodpovedá hmotnosti 2.

vstup	výstup
4 4 1 10 100 1000	11

Všimnite si, že dvojprvková množina obsahujúca prvé a druhé závažie je tentokrát ľahšia ako niektoré jednoprvkové množiny.

vstup	výstup
10 512 7 7 7 7 7 7 7 7 7 7	35

A-III-3 Stavebnica funkcií

Za zadaním súťažnej úlohy nájdete študijný text k nej. Študijný text je identický s textom z krajského kola.

Jednotlivé podúlohy súťažnej úlohy môžete riešiť v ľubovoľnom poradí. Každá podúloha je hodnotená zvlášť. Pri riešení konkrétnej podúlohy môžete využívať:

- *Všetky funkcie definované alebo zostrojené v študijnom texte.*
- *Konštantné funkcie k_b^a (kde k_b^a je funkcia s a vstupmi ktorá vždy vráti hodnotu b).*
- *Funkcie z domáceho kola: mul (násobenie), p (predchodca) a sub (odčítanie, ktoré nepodtečie pod nulu).*
- *Funkcie z krajského kola: pow (umocnenie), sgn (signum) a geq (predikát „väčší alebo rovný“).*
- *Funkciu not (logická negácia) zo vzorových riešení krajského kola.*

Konštrukciu už zostrojených funkcií zbytočne nerozpisujte.

*Upozorňujeme ale, že konštrukcie využívajúce myšlienku z poslednej podúlohy krajského kola (vetvenie, „if“) **je potrebné rozpísať**.*

- **Podúloha A (2 body).** Zostrojte binárne funkcie *min* a *max* počítajúce minimum a maximum.
- **Podúloha B (2 body).** Zostrojte unárnu funkciu *last*, ktorá vráti poslednú cifru čísla na vstupe.

  (Formálne, má byť $\forall n : last(n) = n \bmod 10$.)
- **Podúloha C (3 body).** Zostrojte unárnu funkciu *sqrt*, ktorá vypočíta dolnú celú časť odmocniny.

  Dolná celá časť je najväčšie celé číslo neprevyšujúce presný výsledok. Napr. $sqrt(99) = 9$ a $sqrt(100) = 10$.
- **Podúloha D (3 body).** Fibonacciho čísla sú definované rekurzívne: $F_0 = 0$, $F_1 = 1$ a $\forall n \geq 2 : F_n = F_{n-1} + F_{n-2}$. Zostrojte unárnu funkciu fib, ktorá pre vstup n vráti na výstupe hodnotu F_n.

  Aby sme vám ušetrili prácu, v podúlohe D nemusíte formálne rozpisovať použitia Kompozítora. Presnejšie, vždy, keď chcete nejakú novú funkciu vyrobiť Kompozítorom zo skôr vyrobených funkcií, stačí novú funkciu korektne matematicky definovať. (A samozrejme, naozaj sa danú funkciu musí dať vyrobiť zložením iných funkcií, ktoré už máte vyrobené, nemôžete si len tak vyčarovať úplne nové funkcie.)

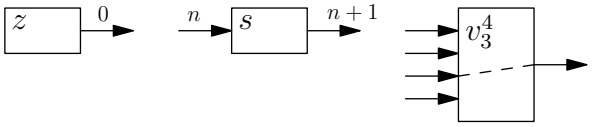
  V riešení tejto podúlohy teda smiete napr. napísať: „Z funkcie *pow* si ľahko vyrobíme funkciu f takú, že $\forall n : f(n) = (n + 4)^7$.“

Študijný text: stavebnica funkcií

Miška dostala na narodeniny zvláštny darček: stavebnicu funkcií. Keď darček rozbalila, našla v ňom hneď niekoľko rôznych vecí. Ako prvé jej oko padlo na tri sáčky s hotovými funkciami. Každá funkcia je malá škatuľka, ktorá má niekoľko vstupov a práve jeden výstup.

- V prvom sáčku bola jediná funkcia. Volala sa z (z anglického „zero“, čiže nula), nemala žiadne vstupy a na výstupe vracala číslo 0.
- Aj v druhom sáčku bola len jedna funkcia. Táto sa volala s (z anglického „successor“, čiže nasledovník). Mala jeden vstup a jeden výstup. Keď na vstup dostala číslo n, vrátila nám na výstupe číslo $n + 1$.
- Tretí sáčok bol o čosi plnší – bolo v ňom nekonečne veľa funkcií. Pre každé k a n (také, že $1 \leq k \leq n$) tam bola funkcia v_k^n („vyber k-ty z n vstupov“), ktorá mala n vstupov a na výstup vždy vrátila hodnotu, ktorú dostala na k-tom vstupe.

Na obrázku sú znázornené funkcie z, s a v_3^4.



Zvyšok balíčka tvorili dva prístroje, ktoré zjavne slúžia na výrobu nových funkcií. Na jednom z nich sa skvel nápis Kompozítor, na druhom zase Cyklovač. Každý z nich funguje tak, že do vnútra v správnom poradí vložíme nejaké funkcie, zatočíme kľukou a vypadne nám nová funkcia. Tá je vhodne poskladaná z funkcií, ktoré sme vložili dovnútra. Ale skôr ako si podrobnejšie popíšeme fungovanie Kompozítora a Cyklovača, potrebujeme si o naších funkciách povedať čosi formálnejšie.

V tejto úlohe považujeme nulu za prirodzené číslo. Prirodzené čísla sú teda pre nás množina $\mathbb{N} = \{0, 1, 2, 3, \dots\}$. Všetky funkcie, ktoré si budeme vyrábať, budú totálne funkcie na prirodzených číslach. Ako vstupy budeme teda funkcii dávať prirodzené čísla a pre každý možný vstup nám funkcia vráti na výstup jedno prirodzené číslo. Počet vstupov funkcie sa nazýva *arita*. Napr. funkcie s jedným vstupom odborne voláme *unárne*, funkcie s dvoma vstupmi *binárne*, atď. Funkcia v_2^7 má aritu 7. Funkcia z má aritu 0. Aritu občas budeme explicitne písať ako horný index. Mohli by sme teda napr. hovoriť, že v prvom sáčku bola funkcia z^0 a v druhom zas funkcia s^1. (U vyberacích funkcií v_k^n aritu musíme uvádzať vždy, keďže napr. v_1^4 a v_1^7 sú dve rôzne funkcie.)

Akonáhle už nejakú funkciu vyrobíme, máme navždy k dispozícii ľubovoľne veľa jej kópií. Špeciálne platí, že ak funkciu použijeme pri výrobe inej, zložitejšej, nestratíme ju tým.

Kompozítor

Kompozítor vie funkcie skladať (v bežnom matematickom zmysle tohto slova). Je však trochu háklivý na správne arity. Použitie Kompozítora sa skladá z nasledujúcich krokov:

1. Vyberieme si aritu $a \geq 0$ funkcie, ktorú chceme vyrobiť.
2. Zvolíme si funkciu g^b (čiže funkciu g s nejakou aritou b, možno inou ako a), ktorú použijeme v druhom kroku výpočtu.

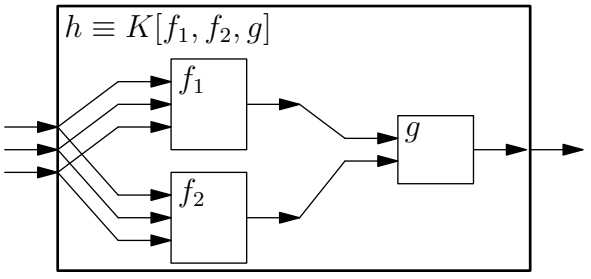
   Hodnota b musí byť kladná. (Inými slovami, funkcia g musí mať aspoň jeden vstup.)
3. Zvolíme si b funkcií $f_1^a, \dots, f_b^a$, ktoré použijeme v prvom kroku výpočtu.
4. Zatočíme kľukou na boku Kompozítora. Na výstupe nám vypadne nová funkcia h definovaná nasledujúcim pseudokódom:

```
def h ( x_1, ..., x_a ):
    tmp_1 = f_1 ( x_1, ..., x_a )
    tmp_2 = f_2 ( x_1, ..., x_a )
    ...
    tmp_b = f_b ( x_1, ..., x_a )

    return g ( tmp_1, ..., tmp_b )
```

   Slovne: Funkcia h zoberie a vstupov, ktoré dostala. Pomocou funkcií f_1 až f_b z nich vypočíta b pomocných hodnôt. No a na záver pomocou funkcie g vypočíta z pomocných hodnôt výstup celej funkcie h.

Na obrázku je graficky znázornená funkcia vyrobená Kompozítorom pre $a = 3$ a $b = 2$.



Cyklovač

Cyklovač vie vyrábať for-cykly. Aj on je však trochu háklivý na správne arity funkcií a poradie ich parametrov. Dajte si na ne pozor, keď ho budete kŕmiť. Správne použitie Cyklovača vyzerá nasledovne:

1. Vyberieme si aritu $a \geq 1$ funkcie, ktorú chceme vytvoriť.

   Prvý parameter tejto funkcie (označíme ho x) bude špeciálna premenná, ktorá hovorí, koľkokrát sa má for-cyklus vykonať. Ostatné parametre (označíme ich $y_1, \ldots, y_{a-1}$) budú ostávať nezmenené.

2. Zvolíme si funkciu f^{a-1}, ktorou výpočet cyklu začne.

3. Zvolíme si funkciu g^{a+1}, ktorá počíta, čo sa stane počas jednej iterácie cyklu. Funkcia g má až $a+1$ vstupov: všetky premenné, ktoré bude mať aj výsledná funkcia a navyše hodnotu ktorá bola výstupom predchádzajúcej iterácie cyklu. (Ak to nedávalo zmysel, pozri pseudokód uvedený v ďalšom kroku.)

4. Zatočíme kľukou na boku Cyklovača. Na výstupe nám vypadne nová funkcia h definovaná nasledujúcim pseudokódom:

   ```
   def h ( x, y_1, ..., y_{a-1} ):
       tmp = f ( y_1, ..., y_{a-1} )
       for i = 0 to x-1:
           tmp = g ( i, y_1, ..., y_{a-1}, tmp )
       return tmp
   ```

   Slovne: Výpočet začne tým, že funkciou f vypočítame (z ostatných parametrov) výstup pre $x = 0$. Z neho potom funkciou g vypočítame výstup pre $x = 1$, z toho znova funkciou g výstup pre $x = 2$, a tak ďalej až po požadovanú hodnotu prvého parametra.

Notácia

Rovnosť dvoch funkcií budeme značiť $\equiv$. Zápis $f \equiv g$ teda znamená, že funkcie f a g majú rovnakú aritu a na každom vstupe dávajú ten istý výstup.
Funkciu ktorá vznikne Kompozítorom z funkcií $f_1, \ldots, f_b$ a g budeme značiť $K[f_1, \ldots, f_b, g]$.
Funkciu ktorá vznikne Cyklovačom z funkcií f a g budeme značiť $C[f, g]$.

Príklady

Uf, to vyzerá komplikovane. Poďme sa preto spolu pozrieť na to, ako si Miška začala vyrábať nejaké nové funkcie.

Príklad 1. Čím by sme tak začali? Peknou jednoduchou funkciou je napríklad identita: unárna funkcia i taká, že pre každé n je $i(n) = n$. Vidíte, ako ju vyrobiť?
Toto bola triková otázka. Identitu vyrábať nepotrebujeme, dostali sme ju totiž v treťom sáčku. Identitou je totiž funkcia v_1^1. Môžeme teda písať $i \equiv v_1^1$.

Príklad 2. Vyrobme si funkciu j^0: konštantnú funkciu, ktorá nemá žiadny vstup a na výstupe vracia hodnotu 1. Túto funkciu si vieme vyrobiť Kompozítorom. V prvom kroku použijeme funkciu z, ktorá nemá žiaden vstup a na výstupe vráti 0. Túto 0 potom „pošleme ďalej“ do funkcie s, ktorá ju zväčší na 1.
Dostávame teda, že $j^0 \equiv K[z, s]$.

Príklad 3. Unárnu funkcia $plus3$, ktorá svoj jediný vstup zväčší o 3, vieme vyrobiť napríklad ako $K[K[s, s], s]$. Teda najskôr si vyrobíme funkciu $K[s, s]$, ktorá svoj vstup zväčší o 2, a túto potom opäť vložíme do Kompozítora s ďalšou funkciou s.

Príklad 4. Teraz si ukážeme, ako si Miška vie vyrobiť sčítanie – teda binárnu funkciu add takú, že $\forall x, y : add(x, y) = x + y$.
Základným pozorovaním je, že sčítanie je vlastne opakované použitie funkcie „+1“, teda nasledovníka. Výpočet $x + y$ si teda môžeme sformulovať nasledovne: „začni s hodnotou y a potom na ňu x-krát použi funkciu s“. No a keďže toto vyzerá ako cyklus, na výrobu sčítania budeme chcieť použiť Cyklovač.

Pozrime sa teda na pseudokód funkcie, ktorú vyrobí Cyklovač (s tým, že si ho už upravíme konkrétne na funkciu s dvoma vstupmi).

```
def add(x,y):
    tmp = f(y)
    for i = 0 to x-1:
        tmp = g(i,y,tmp)
    return tmp
```

Aké funkcie f a g potrebujeme vhodiť do Cyklovača ak chceme dostať program pre sčítanie?
Funkcia f má jednoducho vrátiť hodnotu y, ktorú dostala na vstupe – čiže f má byť identita.
Funkcia g má v každej iterácii cyklu zobrať starú hodnotu (uloženú v premennej `tmp`) a inkrementovať ju použitím funkcie s. Potrebujeme teda funkciu s tromi vstupmi, ktorá prvé dva odignoruje, na tretí použije s a vráti výsledok. Takúto funkciu síce ešte nemáme, ale vieme si ju ľahko vyrobiť Kompozítorom: je to funkcia $K[v_3^3, s]$.
Dokopy teda dostávame, že $add \equiv C[\, v_1^1,\, K[v_3^3, s]\,]$.

Príklad 5. Ďalšou jednoduchou funkciou je unárna konštantná nula, teda funkcia zz^1, ktorá má jeden vstup a na výstupe vždy vracia nulu. (Formálne: $\forall n : zz^1(n) = 0$.)
Skôr, než si ukážeme, ako zz^1 vyrobiť, podotknime, že zz^1 a z^0 (čo je funkcia z, ktorú sme dostali v prvom sáčku) sú dve rôzne funkcie.
Zdalo by sa, že zz^1 musí ísť nejak vyrobiť zo z^0 pomocou Kompozítora. Lenže ako? Ak by sme chceli z^0 použiť v prvom kroku, tak bez ohľadu na to, akú funkciu použijeme v druhom kroku, určite vyrobíme funkciu s aritou 0. No a v druhom kroku z^0 použiť nesmieme, lebo funkcia použitá v druhom kroku musí mať kladnú aritu.
Cez Kompozítor teda cesta nevedie. Ukážeme si však, že zz^1 vieme vyrobiť pomocou Cyklovača. Opäť začnime tým, že sa pozrieme, ako to vyzerá, keď chceme pomocou Cyklovača vyrobiť unárnu funkciu. My dodáme funkcie f^0 a g^2 a Cyklovač nám z nich vyrobí funkciu h^1 definovanú nasledovne:

```
def h(x):
    tmp = f()
    for i = 0 to x-1:
        tmp = g(i,tmp)
    return tmp
```

Ako máme zvoliť funkcie f a g, ak chceme, aby h na každom vstupe vracala nulu? Zjavne musíme zvoliť $f \equiv z^0$, aby bolo $h(0) = 0$. No a jedna možnosť ako teraz zvoliť funkciu g je jednoducho zobrať $g \equiv v_2^2$. Tým sa z príkazu `tmp = g(i,tmp)` stane príkaz `tmp = tmp`, a teda v premennej `tmp` ostane stále nula, bez ohľadu na hodnotu x.
Touto pomerne obskurnou konštrukciou sme si teda ukázali, že funkciu zz^1 vieme zostrojiť ako $C[z, v_2^2]$.

Predikáty. Predikát je odborné meno pre funkciu, ktorej návratová hodnota je logická hodnota: pravda alebo nepravda. Naše funkcie takéto hodnoty síce vracať nevedia, pomôžeme si ale rovnako ako voľakedy napríklad autori programovacieho jazyka C: prehlásime 0 za nepravdu a 1 za pravdu.
Funkciu teda budeme volať predikát vtedy, ak:

- táto funkcia na každom možnom vstupe vráti jednu z hodnôt 0 a 1
- chceme zdôrazniť, že na tieto hodnoty sa má zmysel dívať ako na logické hodnoty namiesto číselných

Príklady: Funkcia $eq(x, y)$, ktorá vráti 1 ak sa jej vstupy rovnajú a 0 inak, je predikát. Predikátom je aj funkcia $isprime(x)$ ktorá vráti hodnotu 1 ak x je prvočíslo a 0 inak. Na unárnu konštantnú funkciu zz^1 sa môžeme dívať ako na predikát, ktorý vždy vracia hodnotu „nepravda“.

TRIDSIATY TRETÍ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Eduard Batmendijn, Michal Forišek, Samuel Gurský, Samuel Sládek, Emanuel Tesař
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2018