Informácie a pravidlá

Pre koho je súťaž určená?

Do **kategórie B** sa smú zapojiť len tí žiaci základných a stredných škôl, ktorí ešte ani v tomto, ani v nasledujúcom školskom roku nebudú končiť strednú školu.
Do **kategórie A** sa môžu zapojiť všetci žiaci (základných aj) stredných škôl.

Odovzdávanie riešení domáceho kola

Riešitelia domáceho kola odovzdávajú riešenia sami, v elektronickej podobe, a to priamo na stránke olympiády: http://oi.sk/. Odovzdávanie riešení bude spustené niekedy v septembri.

Riešenia kategórie A je potrebné odovzdať najneskôr **15. novembra 2017**.
Riešenia kategórie B je potrebné odovzdať najneskôr **1. decembra 2017**.

Priebeh súťaže

Za každú úlohu domáceho kola sa dá získať od 0 do 10 bodov. Na základe bodov domáceho kola stanoví Slovenská komisia OI (SK OI) pre každú kategóriu bodovú hranicu potrebnú na postup do **krajského kola**. Očakávame, že táto hranica bude približne rovná **tretine maximálneho počtu bodov**.

V krajskom kole riešitelia riešia štyri teoretické úlohy, ktoré môžu tematicky nadväzovať na úlohy domáceho kola. V kategórii B súťaž týmto kolom končí.

V kategórii A je približne najlepších 30 riešiteľov krajského kola (podľa počtu bodov, bez ohľadu na kraj, v ktorom súťažili) pozvaných do **celoštátneho kola**. V celoštátnom kole účastníci prvý deň riešia teoretické a druhý deň praktické úlohy. Najlepší riešitelia sú vyhlásení za víťazov. Približne desať najlepších riešiteľov následne SK OI pozve na týždňové výberové sústredenie. Podľa jeho výsledkov SK OI vyberie družstvá pre Medzinárodnú olympiádu v informatike (IOI) a Stredoeurópsku olympiádu v informatike (CEOI).

Ako majú vyzerať riešenia úloh?

V praktických úlohách je vašou úlohou vytvoriť program, ktorý bude riešiť zadanú úlohu. Program musí byť v prvom rade korektný a funkčný, v druhom rade sa snažte aby bol čo najefektívnejší.

V kategórii B môžete použiť ľubovoľný programovací jazyk.

V kategórii A musíte riešenia praktických úloh písať v jednom z podporovaných jazykov (napr. C++, Pascal alebo Java). Odovzdaný program bude automaticky otestovaný na viacerých vopred pripravených testovacích vstupoch. Podľa toho, na koľko z nich dá správnu odpoveď, vám budú pridelené body. Výsledok testovania sa dozviete krátko po odovzdaní. Ak váš program nezíska plný počet bodov, budete ho môcť vylepšiť a odovzdať znova, až do uplynutia termínu na odovzdávanie.

Presný popis, ako majú vyzerať riešenia praktických úloh (napr. realizáciu vstupu a výstupu), nájdete na webstránke, kde ich budete odovzdávať.

Ak nie je v zadaní povedané ináč, riešenia teoretických úloh musia v prvom rade obsahovať **podrobný slovný popis použitého algoritmu, zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t. j. posúdenie časových a pamäťových nárokov programu). Na záver riešenia uveďte program. Ak používate v programe netriviálne algoritmy alebo dátové štruktúry (napr. rôzne súčasti STL v C++), súčasťou popisu algoritmu musí byť dostatočný popis ich implementácie.

Usporiadateľ súťaže

Olympiádu v informatike (OI) vyhlasuje *Ministerstvo školstva SR* v spolupráci so *Slovenskou informatickou spoločnosťou* (odborným garantom súťaže) a *Slovenskou komisiou Olympiády v informatike*. Súťaž organizuje *Slovenská komisia OI* a v jednotlivých krajoch ju riadia *krajské komisie OI*. Na jednotlivých školách ju zaisťujú učitelia informatiky. Celoštátne kolo OI, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje IUVENTA v tesnej súčinnosti so Slovenskou komisiou OI.

A-I-1 Trojice

*Toto je **praktická úloha**. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**.*

Keď bol naposledy Mário chorý, strašne sa doma nudil. Jedno ráno si preto vyrobil n kartičiek a na každú napísal nejaké prirodzené číslo. Potom sa postupne pozrel na všetky možné trojice kartičiek. Pre každú trojicu spočítal súčet čísel na nich a výsledok si zapísal na papier.

Súťažná úloha

Dané je n a zoznam čísel na kartičkách. Ďalej je dané **rozumne malé** prirodzené číslo k. Nájdite k-te najmenšie číslo na Máriovom papieri – čiže k-ty najmenší spomedzi všetkých súčtov trojíc čísel na kartičkách.
Ak sa nejaký súčet dá vyrobiť viacerými spôsobmi, rátame každý jeho výskyt zvlášť. (Viď prvý príklad.)

Formát vstupu a výstupu

V prvom riadku vstupu sú celé čísla n a k. V druhom riadku vstupu je n navzájom rôznych celých čísel $a_1, \ldots, a_n$. Sú usporiadané od najmenšieho po najväčšie.

Vypíšte jeden riadok a v ňom jedno číslo: k-ty najmenší spomedzi všetkých súčtov trojíc.

Obmedzenia a hodnotenie

Je 10 sád testovacích vstupov, za každú môžete získať 1 bod.
Vo všetkých sadách platí $3 \leq n \leq 100\,000$, $1 \leq k \leq 1\,000\,000$, $k \leq \binom{n}{3}$ a $1 \leq a_1 < a_2 < \cdots < a_n \leq 10^{12}$.
(Pozor, na uloženie a_i použite premennú s dostatočným rozsahom!)

V prvých troch sadách platí $n \leq 500$. V ďalších troch sadách platí $n \leq 7000$.

Príklady

vstup

```
5 4
1 2 3 4 5
```

výstup

```
8
```

Všetkých 10 možných súčtov: 6, 7, 8, 8, 9, 9, 10, 10, 11, 12. Teda aj pre $k = 3$ aj pre $k = 4$ je správnym výstupom číslo 8. Pre $k = 9$ by bolo správnym výstupom číslo 11.

vstup

```
5 4
1 10 100 1000 10000
```

výstup

```
1110
```

Tento raz sú všetky súčty trojíc navzájom rôzne.

A-I-2 Telenovela

*Toto je **praktická úloha**. Pomocou webového rozhrania odovzdajte **funkčný, odladený program**.*

V televízii ide nová telenovela. Jaro by ju chcel sledovať, ale má kopec roboty. Ešteže všetky diely tak často reprízujú. A tak Jaro sedí, v jednej ruke svoj kalendár, v druhej televízny program, a húta, ktoré diely si vlastne má pozrieť. O pozeraní telenovely platia nasledovné fakty:

- Úplne na začiatku treba pozrieť prvý diel, v ktorom sa zoznámime s hlavnými postavami.
- Nie je potrebné vidieť všetky diely. Všetko dôležité sa opakuje, takže vieme ľubovoľne veľa dielov vynechať.
- Je ale veľmi dôležité sledovať diely v správnom poradí. Akonáhle si pozriete čo len jeden diel mimo poradia (teda najskôr neskorší a potom skorší), už nikdy sa nezorientujete v tom, čo sa vlastne kedy stalo.
- Pochopiteľne, žiadny diel nechceme sledovať dvakrát.
- Na úplný záver je potrebné vidieť posledný diel, v ktorom príde happyend.

Súťažná úloha

Daná je postupnosť $a_1, \ldots, a_n$ čísel dielov, ktorých vysielanie by Jaro stíhal sledovať. Zistite, či si vie pozrieť telenovelu tak, aby dodržal vyššie uvedené pravidlá. Ak áno, zistite tiež, koľko najviac dielov vie vidieť.

Formát vstupu a výstupu

V prvom riadku vstupu sú celé čísla n (počet dielov, ktoré Jaro stíha sledovať) a e (počet epizód telenovely). Epizódy sú očíslované od 1 po e.
V druhom riadku je postupnosť n celých čísel $a_1, \ldots, a_n$: čísla odvysielaných epizód v chronologickom poradí.

Vypíšte jeden riadok a v ňom jedno číslo: maximálny počet dielov, ktoré vieme vidieť pri správnom pozretí tejto telenovely. Ak správnym spôsobom telenovelu pozrieť nevieme, vypíšte namiesto toho číslo -1.

Obmedzenia a hodnotenie

Je 10 sád testovacích vstupov, za každú môžete získať 1 bod.
Vo všetkých sadách platí $1 \leq n \leq 100\,000$, $2 \leq e \leq 10^9$ a pre každé i platí $1 \leq a_i \leq e$.
Rôzne sady majú rôzne veľkú (postupne rastúcu) maximálnu hodnotu n. V druhej sade je $n = 10$, v tretej sade je $n = 14$, v piatej sade je $n = 100$, v siedmej sade je $n = 7000$.
V sadách s nepárnym číslom platí, že v každom testovacom vstupe sú všetky a_i navzájom rôzne.

Príklady

vstup

```
5 50
33 1 50 20 47
```

výstup

```
2
```

Pozrie si prvú a následne poslednú epizódu.

vstup

```
8 50
1 3 3 3 3 50 1 4
```

výstup

```
3
```

Pozrie si epizódu 1, jedno zo štyroch vysielaní epizódy 3 a na záver epizódu 50.

vstup

```
6 50
47 4 6 3 5 1
```

výstup

```
-1
```

Ak pozrie prvú epizódu, už nestihne žiadnu ďalšiu. A navyše aj tak nikdy neuvidí poslednú epizódu.

vstup

```
6 6
1 2 4 3 5 6
```

výstup

```
5
```

A-I-3 Reverzy prefixov

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách. Každá podúloha je hodnotená samostatne, nie je nutné ich riešiť v uvedenom poradí.

Vo všetkých podúlohách máme na začiatku pole $A[1..n]$, v ktorom je uložená nejaká permutácia čísel 1 až n. Inými slovami, každé z čísel od 1 po n sa v poli A nachádza presne raz.
Toto pole smieme meniť len pomocou operácie $flip(k)$. Hodnota k môže byť od 1 do n, vrátane. Efekt tejto operácie je že sa obráti poradie prvých k prvkov poľa A.
Napríklad z poľa $A = (1, 2, 3, 5, 4)$ operácia $flip(3)$ spraví pole $(3, 2, 1, 5, 4)$.

Podúloha A (2 body)

Nájdite algoritmus s polynomiálnou časovou zložitosťou, ktorý ľubovoľné vstupné pole A usporiada.

Podúloha B (2 body)

Dávid si vymyslel, že bude do nekonečna opakovať operáciu $flip(A[1])$. V každom kroku sa teda pozrie na prvý prvok poľa A a následne otočí poradie toľkých prvkov, akú hodnotu uvidel. Napríklad ak na začiatku poľa uvídí číslo 4, spraví $flip(4)$.
Dávidovi sa zdá, že nech začne od akéhokoľvek poľa, vždy sa časom stane, že sa na začiatok poľa dostane číslo 1. Tým samozrejme všetka zábava končí, keďže od tej chvíle už bude Dávid donekonečna opakovať operáciu $flip(1)$, ktorá pole vôbec nezmení.
Overte túto hypotézu pre $n = 10$. Zistite, pre ktorú z $10!$ permutácií spraví Dávid najviac krokov kým sa na začiatok poľa dostane jednotka.

Podúloha C (3 body)

Pre $n = 47$ nájdite nejakú permutáciu, pre ktorú Dávid spraví veľa krokov, kým nastane $A[1] = 1$.
Za permutáciu, pre ktorú spraví aspoň 500 krokov, dostanete 1 bod.
Za permutáciu, pre ktorú spraví aspoň 650 krokov, dostanete 2 body.
Za permutáciu, pre ktorú spraví aspoň 750 krokov, dostanete 3 body.

Podúloha D (3 body)

Naozaj platí, že pre úplne každé n a úplne každú permutáciu čísel v poli A sa musí časom na začiatok poľa A dostať číslo 1? Dokážte toto tvrdenie alebo nájdite protipríklad.

A-I-4 Stavebnica funkcií

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách. K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Odporúčame najskôr prečítať ten a až potom sa vrátiť k samotným súťažným úlohám.

Jednotlivé podúlohy súťažnej úlohy môžete riešiť v ľubovoľnom poradí. Každá podúloha má hodnotu 2 body. Požadovanú funkciu smiete zostrojiť „na viac krokov“ – teda najskôr si vyrobiť nejaké jednoduchšie pomocné funkcie.

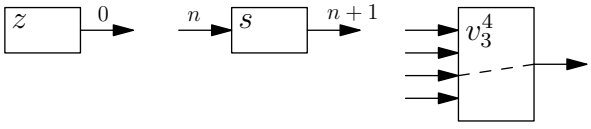
- **Podúloha A.** Zostrojte funkciu zzz^3: funkciu s tromi vstupmi, ktorá vždy vráti na výstup nulu.
  Formálne, má platiť $\forall a, b, c : zzz^3(a, b, c) = 0$.
- **Podúloha B.** Miška už zostrojila nejakú funkciu φ^4. Túto funkciu máte k dispozícii, ale neviete nič o tom, čo počíta. Miška by teraz chcela funkciu ψ^3 definovanú nasledovne: $\forall a, b, c : \psi^3(a, b, c) = \varphi^4(b, a, c, a)$. Ukážte, ako si má Miška vyrobiť túto funkciu.
- **Podúloha C.** Zostrojte násobenie: binárnu funkciu mul takú, že pre všetky a, b platí $mul(a, b) = a \cdot b$.
- **Podúloha D.** Zostrojte predchodcu: unárnu funkciu p takú, že $p(0) = 0$ (lebo záporné čísla nemáme) a $\forall n : p(n + 1) = n$.
- **Podúloha E.** Zostrojte odčítanie, respektíve teda funkciu, ktorá sa naň čo najviac podobá. Presnejšie, zostrojte binárnu funkciu sub takú, že pre $x > y$ je $sub(x, y) = x - y$ a pre $x \leq y$ je $sub(x, y) = 0$.

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.

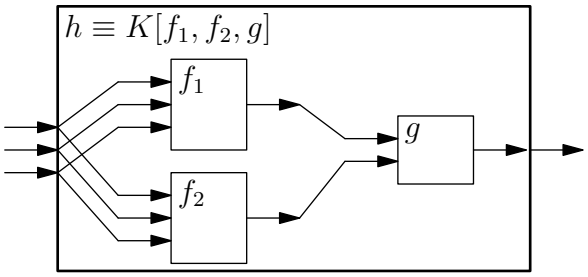
   Hodnota b musí byť kladná. (Inými slovami, funkcia g musí mať aspoň jeden vstup.)
3. Zvolíme si b funkcií $f_1^a, \ldots, f_b^a$, ktoré použijeme v prvom kroku výpočtu.
4. Zatočíme kľukou na boku Kompozítora. Na výstupe nám vypadne nová funkcia h definovaná nasledujúcim pseudokódom:

```
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 $C[z, v_2^2]$.

TRIDSIATY TRETÍ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Michal Forišek
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2017