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. 11. 2021**.
Riešenia kategórie B je potrebné odovzdať najneskôr **30. 11. 2021**.

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 Tancujúci kráľ

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

V rade stojí n tanečníkov. Idúc zľava doprava si pozície v rade očíslujeme od 1 po n. Výšku tanečníka, ktorý začína na pozícii i, označíme h_i. Tanečník číslo k je kráľ.

Chceli by sme všetkých tanečníkov usporiadať podľa výšky (do neklesajúceho poradia).

Aby to dobre vyzeralo, tanečníci si smú meniť poradie len jediným spôsobom: pomocou tanečných figúr. Každá figúra vyzerá tak, že sa nejaký súvislý úsek tanečníkov zatočí a skončí v presne opačnom poradí ako začínal.

Ak máme napríklad tanečníkov s výškami $(180, 185, 183, 182, 181, 185, 190)$ tak ich vieme usporiadať pomocou jedinej tanečnej figúry. Zapoja sa do nej tanečníci na pozíciách 2 až 5.

Je verejným tajomstvom, že kráľ vôbec nevie tancovať. Aby sme ho nestrápnili, je nutné, aby sa nikam nehýbal – pri každej figúre musí kráľ ostať na svojom mieste v rade.

Súťažná úloha

Pre daný popis situácie zistite, či vôbec vieme tanečníkov usporiadať. Ak áno, nájdite nejaký spôsob ako to spraviť rozumne malým počtom tanečných figúr.

V každom vstupe dostanete popis situácie na začiatku a hodnotu ℓ. Ak $\ell = 0$, len zistite, či riešenie existuje. Ak je ℓ kladné, pre riešiteľné vstupy aj zostrojte jedno možné riešenie používajúce nanajvýš ℓ figúr.

(Vo vstupoch s $\ell > 0$ bude pre každý riešiteľný vstup existovať riešenie s nanajvýš ℓ figúrami. Ľubovoľné riešenie spĺňajúce tento limit bude akceptované, netreba minimalizovať počet figúr.)

Formát vstupu a výstupu

V prvom riadku vstupu sú celé čísla n, k a ℓ. V druhom riadku sú čísla $h_1, \ldots, h_n$.

Prvý riadok výstupu má obsahovať reťazec „`ANO`" ak sa tanečníkov dá usporiadať, resp. „`NIE`" ak sa to nedá.

Pre $\ell > 0$ nasleduje ešte druhá časť výstupu. Tá má začínať riadkom obsahujúcim číslo $f \leq \ell$: počet figúr, ktoré chcete spraviť. Zvyšok výstupu potom tvorí f riadkov ktoré popisujú jednotlivé figúry v chronologickom poradí. Pre každú figúru vypíšte riadok tvaru „z_j k_j“, kde $1 \leq z_j \leq k_j \leq n$ sú začiatok a koniec úseku pozícií ktorých poradie vaša tanečná figúra obráti.

Obmedzenia a hodnotenie

Vo všetkých vstupoch platí $1 \leq n \leq 1000$, $1 \leq k \leq n$ a $1 \leq h_i \leq 10^9$ pre všetky i.

Je päť sád vstupov. Za každú sadu vstupov, ktoré tvoj program všetky správne vyrieši, dostaneš dva body. V rámci každej sady vstupov majú všetky vstupy tú istú hodnotu ℓ. V niektorých sadách je navyše zaručené, že všetky vstupy budú mať malé n.

číslo sady	1	2	3	4	5
hodnota ℓ	0	1000	2000	3000	1500
maximálne n	1000	20	100	1000	1000

Príklady

vstup

```
5 3 100
160 165 175 170 180
```

výstup

```
NIE
```

Kráľ stojí v strede. Keďže sa nesmie hýbať, riešenie zjavne neexistuje.

vstup

```
5 3 100
185 180 175 170 165
```

výstup

```
ANO
1
1 5
```

Kráľ opäť stojí v strede. Otočíme celý rad tanečníkov. Všimnite si, že táto figúra obsahuje aj kráľa. Ten ale počas nej ostane na svojom mieste, takže je všetko v poriadku.

vstup

```
5 3 0
175 170 175 180 175
```

výstup

```
ANO
```

Tento vstup má $\ell = 0$, takže len treba povedať, či má alebo nemá riešenie.

vstup

```
5 3 100
175 170 175 180 175
```

výstup

```
ANO
4
4 5
1 2
4 5
4 5
```

A-I-2 Prominencia

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

Prominencia vrcholu (niekedy tiež nazývaná *význačnosť* alebo *relatívna výška*) popisuje, nakoľko tento vrchol vyčnieva z okolitej krajiny. Má viacero navzájom ekvivalentných definícií. Jedna z nich vyzerá nasledovne:

- Prominencia najvyššieho vrcholu na svete je rovná jeho výške.
- Pre každý iný vrchol platí, že ak v je jeho nadmorská výška, tak jeho prominencia je najmenšie také p, že ak začneme túru na našom vrchole a chceme sa dostať na nejaké vyššie položené miesto, musíme niekedy po ceste klesnúť do nadmorskej výšky $v - p$.

Napríklad Kráľova hoľa (1946 m.n.m.) má prominenciu 756 metrov. Ak začneme výlet na Kráľovej holi a nikdy neklesneme na 1946 - 756 = 1190 m.n.m., vieme sa pohybovať len po východnej časti Nízkych Tatier a tam žiaden vyšší vrchol nestretneme. Cez sedlo Priehyba (práve 1190 m.n.m.) sa však vieme dostať na západnú časť hrebeňa a tam časom vystúpať do nadmorskej výšky vyššej ako má Kráľova hoľa. (Napríklad sa tak stane cestou na Chopok alebo Ďumbier.)

Súťažná úloha

Z mora trčí jeden horský hrebeň. Predpokladáme, že to je jediná hora na celom svete. Hrebeň vieme popísať ako lomenú čiaru určenú bodmi so súradnicami $(0, 0), (1, h_1), (2, h_2), \ldots, (n, h_n), (n + 1, 0)$. Prvá súradnica je vodorovná, druhá je zvislá.

Vrcholy sú lokálne maximá hrebeňa: tie jeho body a vodorovné úseky, z ktorých ide hrebeň na obe strany dodola. Pre každý vrchol vypočítajte jeho prominenciu.

Formát vstupu a výstupu

V prvom riadku vstupu je celé číslo n. V druhom riadku sú celé čísla $h_1, \ldots, h_n$.

Na výstup vypíšte pre každý vrchol, v poradí v ktorom ležia na hrebeni, jeden riadok s jeho prominenciou.

Obmedzenia a hodnotenie

Vo všetkých vstupoch platí $1 \leq n \leq 200\,000$ a $\forall i : 1 \leq h_i \leq 10^9$.

Odovzdané riešenie bude otestované na šiestich sadách vstupov. Sú za ne postupne 1, 1, 1, 2, 2 a 3 body. Rôzne sady majú rôznu maximálnu hodnotu n:

číslo sady	1	2	3	4	5	6
maximálne n	20	20	5 000	5 000	200 000	200 000

Navyše v sadách 1, 3 a 5 platí, že všetky hodnoty h_i vo vstupe sú navzájom rôzne.

Príklady

vstup

```
7
47 42 47 42 47 47 42
```

výstup

```
47
47
47
```

V tomto pohorí sú tri vrcholy: na indexe 1, na indexe 3 a na indexoch 5-6. Všetky tri sú najvyššími vrcholmi sveta a teda majú prominenciu rovnú svojej výške.

vstup

```
7
47 42 47 43 48 48 42
```

výstup

```
5
4
48
```

Toto je podobné pohorie, ale tentokrát je jediným najvyšším vrcholom sveta masív na indexoch 5-6. Zvyšné dva vrcholy majú výrazne menšiu prominenciu ako v predošlom príklade.

vstup

```
7
47 43 47 42 48 48 42
```

výstup

```
5
5
48
```

Tentokrát majú vrcholy na indexoch 1 a 3 rovnakú prominenciu – pre oba platí, že ak sa z nich chceme dostať niekam vyššie, treba prejsť cez sedlo na indexe 4 s nadmorskou výškou 42.

vstup

```
10
1955 1840 1937 1837 2004 2024 1232 1660 1190 1946
```

výstup

```
118
97
2024
428
756
```

Zjednodušený kus Nízkych Tatier. Postupne zľava doprava Chabenec (vrchol), sedlo pod Chabencom, Kotliská (vrchol), sedlo Poľany, Dereše, Chopok (vrchol), Čertovica, Homôľka (vrchol), sedlo Priehyba a Králova hoľa (vrchol).

A-I-3 Strelec

Toto je teoretická úloha. Pomocou webového rozhrania odovzdajte súbor vo formáte PDF, obsahujúci riešenie, spĺňajúce požiadavky uvedené v pravidlách.

Strelec je klasická šachová figúrka. Pohybuje sa tak, že ho posunieme v niektorom šikmom smere (po niektorej diagonále) o jedno alebo viac políčok. (Ak máme políčka šachovnice klasickým spôsobom ofarbené čiernou a bielou, bude sa každý strelec pohybovať len po políčkach jednej farby.)

Máme obriu šachovnicu s r riadkami a s stĺpcami. Na každom políčku je napísané jedno písmeno.

Chceme po nej preskákať strelcom tak, aby sa postupne zastavil presne na písmenkách daného slova w. (Začať môžeme na ľubovoľnom políčku obsahujúcom prvé písmeno w, každým ďalším ťahom sa musíme presunúť na políčko obsahujúce jeho nasledujúce písmeno.)

Koľkými rôznymi spôsobmi sa to dá spraviť?

Namiesto správnej odpovede stačí, keď vypočítate, aký zvyšok dáva po delení $10^9 + 7$. (Správna odpoveď totiž môže byť veľmi veľká.)

Formát vstupu a výstupu

V prvom riadku vstupu sú rozmery šachovnice: r a s. V druhom riadku je slovo w. Zvyšok vstupu tvorí popis písmen na šachovnici: r riadkov a v každom z nich s písmen.

Na výstup vypíšte jeden riadok so správnou odpoveďou.

Obmedzenia a hodnotenie

Môžete predpokladať, že všetky písmená na vstupe sú veľké písmená anglickej abecedy. Veľkosť abecedy môžete pri analýze algoritmu považovať za konštantu.

Plný počet bodov môžu dostať riešenia, ktoré efektívne vyriešia vstupy s $r, s \leq 1000$ a $|w| \leq 100$.

Nanajvýš 7 bodov môžu dostať riešenia, ktoré efektívne vyriešia vstupy s $r, s, |w| \leq 100$.

Nanajvýš 5 bodov môžu dostať riešenia, ktoré efektívne vyriešia vstupy s $r, s \leq 6$ a $|w| \leq 12$.

Nanajvýš 3 body môžu dostať riešenia, ktoré efektívne vyriešia vstupy v ktorých je nanajvýš 10^6 platných spôsobov skákania.

Príklady

vstup

```
3 4
OI
MALO
KIVI
OBAL
```

výstup

```
1
```

vstup

```
3 5
BRAT
BANAN
NEMAL
SUPKU
```

výstup

```
0
```

Keďže na šachovnici nemáme žiadne R ani T, je zjavné, že po písmenách slova BRAT sa preskákať nedá.

vstup

```
3 5
UUUUUU
UUUUU
UKSPU
UUUUU
```

výstup

```
224
```

Ak riadky a stĺpce číslujeme od 0 vľavo hore a políčka zapisujeme (riadok, stĺpec), jedna prípustná možnosť skákania je $(1,0) \to (0,1) \to (2,3) \to (1,4) \to (2,3) \to (1,4)$.

vstup

```
6 8
KSP
KSPPSKSP
PPSKSKSP
SKKSKSKP
PPPSKSKK
KKSKSKPP
PPPSKSKP
```

výstup

```
102
```

Jedno prípustné riešenie je začať na K úplne vľavo hore, odtiaľ sa posunúť o tri políčka doprava dodola na S a odtiaľ dve políčka doľava dodola na P. Iné prípustné riešenie začne rovnakým pohybom na S a potom sa vráti o dve políčka doľava dohora na iné P.

A-I-4 Pokazený rover

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.

Podúloha A (2 body).
V lokalite `jama` sú nejaké kamienky. Napíšte ľubovoľný program, po ktorého skončení bude v lokalite `jama` štyrikrát toľko kamienkov ako teraz. (Pomôcka: najkratšie riešenie tejto podúlohy má len dve inštrukcie.)

Podúloha B (2 body).
V lokalitách `A` a `B` máme nejaké neznáme počty kamienkov, ktoré označíme a a b. Lokalita `C` je prázdna. Napíšte program, ktorý porovná a a b. Ak sú si rovné, do lokality `C` uloží kamienok, inak ju nechá prázdnu. Na konci behu programu musia lokality `A` a `B` obsahovať svoje pôvodné počty kamienkov.

Podúloha C (3 body).
Napíšte pre rover makro pre delenie so zvyškom. Toto makro bude mať ako parametre štyri lokality: Prvé dve na začiatku obsahujú nejaké neznáme počty kamienkov, ktoré označíme a a b (pričom môžete predpokladať, že $b > 0$). Tieto isté počty v nich musia byť aj na konci. Druhé dve lokality sú na začiatku prázdne a chceme do nich uložiť celú časť podielu ($a \operatorname{div} b$) a zvyšok po delení ($a \bmod b$).

Podúloha D (3 body).
V lokalitách `A` a `B` máme nejaké neznáme počty kamienkov, ktoré označíme a a b. Môžete predpokladať, že sú oba kladné. Lokalita `C` je prázdna. Napíšte program, ktorý do lokality `C` uloží práve $nsd(a, b)$ kamienkov (kde nsd označuje najväčšieho spoločného deliteľa). V tejto podúlohe je povolené ľubovoľne zmeniť obsah lokalít `A` a `B`.

Študijný text: Pokazený rover

Na Marse máme rover. Voľakedy sme mali veľké plány, no po tom, ako ho zasiahla piesočná búrka a pokazila mu skoro všetky periférie, ostal rover takmer nepoužiteľný. Jediné, čo ešte stále vie robiť, je presúvať sa medzi lokalitami a nosiť z miesta na miesto nejaké kamienky. Našou úlohou v tejto sérii úloh bude naučiť rover robiť aspoň nejaké použiteľné výpočty. **Nebude nám pri tom záležať na časovej zložitosti** programov, len na ich korektnosti.

Do roveru vieme na diaľku nahrať program: konečnú postupnosť inštrukcií. Niektoré inštrukcie môžu mať návestia (labels) – symbolické mená, pomocou ktorých sa na ne vieme odkazovať.

Rover pozná na Marse dve špeciálne lokality: *jedáleň* a *kameňolom*. Pre stručnosť ich budeme označovať `J` a `K`. V jedálni je momentálne práve jeden kamienok, inak je to lokalita ako každá iná. V kameňolome je vždy k dispozícii dostatočne veľa kamienkov.

Každému inému reťazcu má rover priradenú nejakú unikátnu lokalitu na Marse, kam sa dajú ukladať kamienky. Ak nie je povedané ináč, predpokladáme, že všetky takéto lokality sú prázdne – neobsahujú žiadne kamienky.

Jediná inštrukcia, ktorú ešte rover vie vykonávať, vyzerá nasledovne:

1. Príď do lokality `Y`. Spočítaj si do pomocnej premennej, koľko je tam kamienkov.
2. Príď do lokality `X`. Pokús sa nabrať taký počet kamienkov aký máš v pomocnej premennej.
3. Ak sa to podarilo, odnes tieto kamienky do lokality `Z`, tam ich vysyp a pokračuj nasledujúcou inštrukciou.
4. Ak sa to nepodarilo (t.j. v lokalite `X` nie je dosť kamienkov), vráť lokalitu `X` do pôvodného stavu a pokračuj inštrukciou s návestím `N`.

Program, ktorý pošleme roveru, bude teda postupnosťou takýchto inštrukcii. Inštrukciu vrátane návestia budeme zapisovať nasledovne:

```
navestie: prenes X Y Z N
```

V ľudskej reči môžeme jednotlivé parametre tejto inštrukcie čítať nasledovne:

```
prenes (odkial) (kolko) (kam) (co robit ak sa nepodarilo)
```

Ako štvrtý parameter môžeme písať `-` (pomlčku) ak chceme, aby vykonávanie programu aj v prípade neúspechu pokračovalo nasledujúcou inštrukciou.

Lokality `X`, `Y` a `Z` nemusia byť navzájom rôzne. Jediné obmedzenie je, že kameňolom (kde je „nekonečne veľa“ kamienkov) nesmieme použiť ako lokalitu `Y`.

Vykonávanie programu sa skončí, keď sa dostane na neexistujúcu inštrukciu – teda buď keď vykonáme poslednú inštrukciu a máme pokračovať nasledujúcou po nej, alebo keď skočíme na neexistujúce návestie.

Príklad 1: príkazy, ktoré skoro nič nerobia

Čo spraví rover, keď mu dáme príkaz `prenes X X X I`? Spočíta kamienky v lokalite `X`, potom ich všetky naberie, potom ich na tom istom mieste zase všetky vysype a bude pokračovať nasledujúcou inštrukciou. Takáto inštrukcia teda Mars vôbec nezmení.

Čo spraví rover, keď mu dáme príkaz `prenes X Y X I`? Aj tentokrát sa počet kamienkov v žiadnej lokalite nezmení, sú však už dva možné priebehy výpočtu: ak bolo v lokalite `Y` viac kamienkov ako v lokalite `X`, výpočet bude pokračovať inštrukciou `I`.

Príklad 2: vyprázdni lokalitu

Rozmyslite si sami, akým príkazom vieme z lokality odstrániť všetky kamienky.

Riešenie: Na vyprázdnenie lokality `X` môžeme použiť príkaz `prenes X X K -`. Rover spočíta kamienky v lokalite `X`, všetky naberie a vysype ich v kameňolome.

Príklad 3: naučíme rover sčítať

V lokalitách `A` a `B` máme nejaké neznáme množstvá kamienkov, lokalita `C` je prázdna. Chceli by sme v lokalite `C` mať počet kamienkov rovný súčtu lokalít `A` a `B`. Lokality `A` a `B` pritom chceme nechať nezmenené.

Opäť, skôr ako si nižšie prečítate riešenie, skúste si ho sami vymyslieť.

Riešenie: Stačí príslušné počty kamienkov presypať z kameňolomu do lokality `C`. Jedným korektným riešením je teda nasledovný program:

```
prenes K A C -
prenes K B C -
```

Príklad 4: naučíme rover odčítať

V lokalitách `A` a `B` máme nejaké neznáme počty kamienkov, ktoré označíme a a b. Lokalita `C` je prázdna. Chceli by sme v lokalite `C` mať počet kamienkov rovný $a - b$, resp. rovný nule ak $b > a$. Lokality `A` a `B` pritom chceme nechať nezmenené.

(Na rozdiel od inštrukcie `prenes`, ktorá odoberie kamienky len ak vie odobrať všetky, chceme pri našom odčítaní vždy odobrať koľko ide.)

Riešenie: Do lokality `C` dáme a kamienkov a potom sa pokúsime odobrať b. Ak sa nám to podarí, sme hotoví, ak nie, tak ešte celú lokalitu `C` vyprázdnime do kameňolomu.

Program:

```
          prenes K A C -
          prenes C B K nulovanie
          prenes prazdna J K koniec
nulovanie: prenes C C K
```

Všimnite si, že v treťom kroku (ktorý sa vykoná, ak sme z lokality `C` úspešne odobrali b kamienkov) sa pokúsime z prázdnej lokality presunúť jeden kamienok. To sa nám zaručene nepodarí, program teda skočí na neexistujúce návestie „koniec“ a tým skončí. Štvrtá inštrukcia sa teda vykoná len vtedy, ak na ňu skočíme z druhej.

Príklad 5: naučíme rover násobiť

V lokalitách A a B máme nejaké neznáme počty kamienkov, ktoré označíme a a b. Lokalita C je prázdna. Chceli by sme v lokalite C mať počet kamienkov rovný $a \cdot b$. Lokality A a B pritom chceme nechať nezmenené.

Toto už nevieme spraviť v konštantnom čase. Násobenie je však len opakované sčítanie: kôpku ab kamienkov vyrobíme tak, že na ňu a-krát prinesieme b kamienkov.

Program:

```
        prenes K A Akopia -
cyklus: prenes Akopia J K koniec
        prenes K B C -
        prenes prazdna J K cyklus
```

Na začiatku si vyrobíme kópiu lokality A, ktorú potom počas výpočtu zničíme. Dokola opakujeme: zober jeden kamienok z kópie lokality Akopia a pridaj b kamienkov do lokality C.

Makrá

Roveru sa dajú definovať makrá: zoberieme postupnosť príkazov a dáme jej symbolické meno, aby sme nemuseli tú istú postupnosť príkazov rozpisovať viackrát. Makro môže mať parametre: pri rôznych použitiach môže robiť tie isté inštrukcie ale pre iné lokality a iné návestia.

Vo vnútri makra môžeme používať aj pomocné lokality. Lokality, ktoré dáme pri jeho definícii do hranatých zátvoriek, budú pri každom použití makra nahradené inou lokalitou, ktorá sa nikde inde v programe nepoužíva.

Toto isté sa automaticky stane aj so všetkými návestiami, ktoré dáme inštrukciám v definícii makra. Každé takéto návestie je teda akoby viditeľné len z jedného konkrétneho použitia makra.

Pri definícii makra smieme používať aj iné makrá, ktoré sme už skôr zadefinovali.

Nižšie uvádzame niekoľko definícií makier. Riadky začínajúce mriežkou sú komentáre.

```
# instrukcia, ktora nic nezmeni, len jeden krok caka
MAKRO cakaj
    prenes J J J -
END

# makro s dvoma parametrami-lokalitami: do Y prida tolko kamienkov, kolko je v X
MAKRO pridaj X Y
    prenes K X Y -
END

# makro s jednym parametrom-navestim: skoci na dane navestie
MAKRO skoc N
    prenes [nova_prazdna_lokalita] J K N
END

# makro pre nasobenie: do Z prida sucin poctov kamienkov z X a Y
MAKRO vynasob X Y Z
            pridaj X [Xkopia]
    cyklus: prenes [Xkopia] J K koniec
            prenes K Y Z -
            skoc cyklus
    koniec: cakaj
END
```

Všimnite si, že keď sme vyššie písali samostatný program pre násobenie, stačilo nám, že návestie koniec nikde neexistuje a skokom naň sme ukončili program. Pri písaní makra toto návestie dáme pred prázdnu inštrukciu na konci makra, keďže chceme, aby po ukončení násobenia program ešte pokračoval ďalej.

Príklad 5: naučíme rover počítať tretiu mocninu

Pomocou vyššie uvedených makier ľahko napíšeme program počítajúci tretiu mocninu. V lokalite A máme nejaký neznámy počet kamienkov, ktorý označíme a. Lokalita B je prázdna a chceme do nej uložiť a^3 kamienkov.

Program:

```
vynasob A A pomocna_lokalita
vynasob A pomocna_lokalita B
```

Pre názornosť ešte ukážeme, ako by mohol vyzerať ten istý program, ak by sme všetky makrá nahradili ich definíciami.

```
            prenes  K        A                Akopia1          -
   cyklus1: prenes  Akopia1  J                K                koniec1
```

```
          prenes  K        A                pomocna_lokalita  -
          prenes  hocico1  J                K                 cyklus1
koniec1:  prenes  J        J                J                 -
          prenes  K        A                Akopia2           -
cyklus2:  prenes  Akopia2  J                K                 koniec2
          prenes  K        pomocna_lokalita B                 -
          prenes  hocico2  J                K                 cyklus2
koniec2:  prenes  J        J                J                 -
```

TRIDSIATY SIEDMY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Michal Forišek
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2021