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.

Termíny 42. ročníka OI

- Riešenia domáceho kola sa dá v kategórii A odovzdať do **15. 11. 2026**, v kategórii B do **30. 11. 2026**.
- Krajské kolo oboch kategórií prebehne 19. 1. 2027 zvlášť v každom kraji.
- Celoštátne kolo kategórie A sa bude konať v dňoch 17.-20. 3. 2027 (súťaží sa vo štvrtok a piatok).
- Detailnejšie informácie o priebehu súťaže nájdete na webe OI: https://oi.sk/?s=pravidla

Odovzdávanie riešení domáceho kola

Riešitelia domáceho kola odovzdávajú riešenia sami, v elektronickej podobe, a to priamo na stránke olympiády: http://oi.sk/. Na postup na krajské kolo je potrebné získať aspoň **10 bodov**.

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++, Python alebo Java). Zoznam podporovaných jazykov vrátane konkrétnych verzií ich kompilátorov a interpreterov nájdete v systéme na odovzdávanie riešení. Tiež tam nájdete presnejší popis toho, ako majú vaše programy vyzerať, napr. detaily realizácie vstupu a výstupu.

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.

Ak nie je v zadaní povedané ináč, riešenia **teoretických úloh** musia obsahovať:

- podstatné časti algoritmu ako program (v ľubovoľnom bežnom prog. jazyku) alebo ekvivalentný pseudokód
- podrobný slovný popis použitého algoritmu a zdôvodnenie jeho správnosti
- diskusiu o efektivite zvoleného riešenia (odhad časovej a pamäťovej zložitosti)

Ak používate v programe netriviálne algoritmy alebo dátové štruktúry (napr. rôzne súčasti STL v C++), súčasťou slovného popisu algoritmu by mal byť stručný popis ich implementácie.

Používanie AI v rámci súťaže

Je **zakázané** využívať nástroje umelej inteligencie (AI), napr. veľké jazykové modely, na priame riešenie alebo implementáciu súťažných úloh. Všetky odovzdané výstupy (program aj slovný popis riešenia) musia byť plne dielom súťažiaceho.

Nástroje AI je možné využívať na štúdium a vysvetľovanie všeobecne známych algoritmov a programovacích techník. Napríklad, je v poriadku vyhľadať si pomocou AI informácie o „efektívnych algoritmoch na hľadanie najkratšej cesty“ a nechať si vysvetliť ich princíp alebo implementáciu. Nie je však dovolené použiť AI na hľadanie myšlienky ani na implementáciu riešenia konkrétnej súťažnej úlohy.

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 NIVAM v tesnej súčinnosti so Slovenskou komisiou OI.

B-I-1 Túra

Toto je praktická úloha. Hodnotia sa v nej iba výsledky vašich programov na vopred pripravených testovacích dátach. Detailnejšie pokyny k odovzdávaniu riešení sú uvedené nižšie.

Krištof chce ísť na túru do Nízkych Tatier. V Tatrách je n *zaujímavých miest* (vrchy, studničky, pamätné kríže ...), ktoré by mohol navštíviť. Tieto miesta si očísloval od 1 po n. K jednotlivým zaujímavým miestam je dobrý prístup po cestách, po samotných *turistických chodníkoch* sa však dá ísť iba od zaujímavého miesta i k zaujímavému miestu $i + 1$ a len v tomto smere.

Pri plánovaní túry by mu pomohlo vedieť, v akej nadmorskej výške sa jednotlivé zaujímavé miesta nachádzajú. Takúto informáciu však na internete nenašiel. Podarilo sa mu ale dozvedieť, aké prevýšenie majú cesty medzi každými dvoma za sebou idúcimi miestami. Presnejšie, našiel čísla $v_1, \ldots, v_{n-1}$, kde číslo v_i označuje zmenu nadmorskej výšky na trase od i-teho miesta po $i + 1$-vé miesto. Ak je číslo v_i kladné, znamená to, že trasa stúpa, ak je číslo v_i záporné, tak trasa klesá.

Navyše vie, že zaujímavé miesto číslo 1 sa nachádza vo výške 0 m.n.m.

Súťažná úloha

Zo zadaných informáciách o prevýšeniach by chcel Krištof naplánovať svoju túru a zistiť niečo o výškach jednotlivých zaujímavých miest. Úloha má dve podúlohy:

a) Zistite, koľko zaujímavých miest sa nachádza vo výške 1047 m.n.m.

b) Krištof chce na túre navštíviť niekoľko za sebou idúcich zaujímavých ciest. Začínať vie na ľubovoľnom zaujímavom mieste, z ktorého prejde po niekoľkých turistických chodníkoch a svoju túru ukončí opäť na zaujímavom mieste.

   Chcel by však, aby celkové prevýšenie, ktoré počas takejto túry spraví, bolo rovné 0. Zistite, koľko možností na takúto túru existuje.

   Formálnejšie, nájdite počet takých dvojíc začiatočného miesta i a koncového miesta j, že $1 \leq i < j \leq n$ a $v_i + v_{i+1} + \cdots + v_{j-1} = 0$

Formát vstupu a výstupu

Prvý riadok vstupu obsahuje číslo n – počet zaujímavých miest.

Na druhom riadku je $n - 1$ medzerou oddelených celých čísel $v_1, v_2, \ldots v_{n-1}$. Číslo v_i udáva zmenu nadmorskej výšky medzi miestom i a $i + 1$.

Tretí riadok obsahuje jedno číslo q určujúce úlohu, ktorú máte riešiť.

Na výstup vypíšte jedno celé číslo. Ak je $q = 1$, toto číslo je počet zaujímavých miest vo výške 1047 m.n.m. Ak je $q = 2$, toto číslo udáva počet rôznych túr, ktorých celkové prevýšenie je nulové.

Obmedzenia a hodnotenie

Táto úloha má 5 hodnotených vstupov. Za každý správne vyriešený vstup dostanete dva body.

Vo všetkých vstupoch platí, že hodnoty v_i v absolútnej hodnote nepresiahnu 10 000.

Pre jednotlivé vstupy navyše platia nasledovné podmienky:

vstup	**01.txt**	**02.txt**	**03.txt**	**04.txt**	**05.txt**
q	1	1	2	2	2
n	5000	200 000	100	5000	200 000

Príklad

vstup

```
9
875 366 -194 500 -306 -800 606 194
1
```

výstup

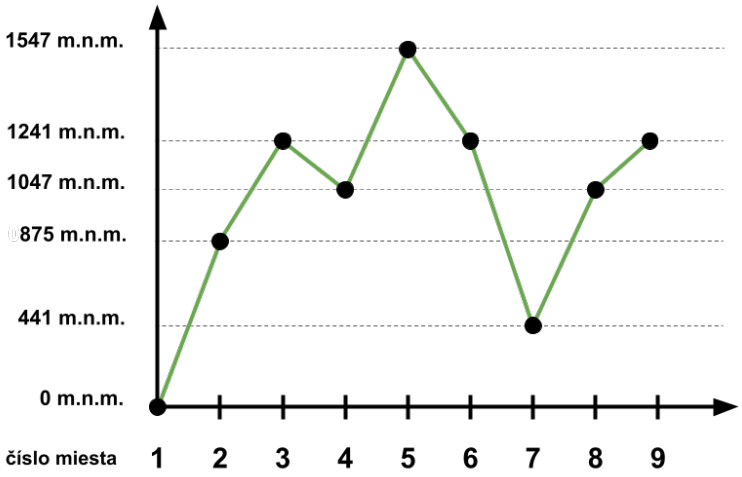
```
2
```

vstup

```
9
875 366 -194 500 -306 -800 606 194
2
```

výstup

```
4
```



Obrázok vyššie znázorňuje zaujímavé miesta popísané v oboch ukážkových vstupoch.
Pre podúlohu a) je odpoveď 2, keďže existujú dve zaujímavé miesta vo výške 1047 m.n.m.
Pre podúlohu b) je odpoveď 4. Nulové prevýšenie totiž dosiahneme pre začiatok 3 a koniec 6 (súčet prevýšení je $-194 + 500 - 306 = 0$), začiatok 4 a koniec 8 (súčet $500 - 306 - 800 + 606$), začiatok 6 a koniec 9 (súčet $-800 + 606 + 194$) a začiatok 3 a koniec 9 (súčet $-194 + 500 - 306 - 800 + 606 + 194$)

Odovzdávanie riešení

Zo stránky `https://oi.sk/?d=rocnik` si stiahnite ZIP archív obsahujúci 5 testovacích vstupov.
Vstupy sú nazvané `01.txt` až `05.txt`.
Úlohu viete odovzdávať dvoma spôsobmi:

1. K čo najviac vstupom vytvoríte správne výstupy a uložíte ich do súborov `sol01.txt` až `sol5.txt`. Následne odovzdáte ZIP archív obsahujúci **zdrojový kód vášho programu** a tieto výstupné súbory.
2. Odovzdáte program v jednom z povolených programovacích jazykov (napr. `Python` alebo `C++`), ktorý číta štandardný vstup a vypisuje na štandardný výstup. Tento program bude automaticky otestovaný na tej istej sade vstupov, ktorá sa nachádza v ZIP archíve. Predlohu toho, ako má vyzerať takýto váš program, nájdete na stránke.

Za každý správny výstupný súbor získate 2 body.

B-I-2 Rieka

Toto je praktická úloha. Hodnotia sa v nej iba výsledky vašich programov na vopred pripravených testovacích dátach. Detailnejšie pokyny k odovzdávaniu riešení sú uvedené nižšie.

V Absurdistane býva veľké sucho. Rieky vysychajú a nad hladinu sa vynárajú plytčiny, kamene, vraky lodí, zrútené mosty a hromada ďalších vecí. Kým je hladina vysoko, preteká všetka voda v rieke ako jeden súvislý prúd. Akonáhle však hladina klesne natoľko, že sa niektorá z prekážok vynorí, rieka sa v tom mieste pretrhne a rozpadne na niekoľko samostatných ramien.

Obyvateľov teraz zaujíma, ako sa na konkrétnom úseku vysychajúcej rieky bude meniť počet jej ramien.

Súťažná úloha

Uvažujme jeden prierez koryta rieky, ktorý je rozdelený na n stĺpcov. Pre každý stĺpec je známa jeho počiatočná hĺbka dna pod hladinou vody (celé číslo od 1 po n).

Hladina vody v rieke postupne klesá, vždy o 1 jednotku. Celkovo voda klesne n-krát, a následne už bude určite celá vysušená. Po klesnutí hladiny o k jednotiek sa v stĺpci s počiatočnou hĺbkou k práve vynorilo dno a voda cezeň už nepreteká.

Ramenom označujeme každý súvislý úsek nevyschnutých stĺpcov prierezu (také, cez ktoré ešte preteká voda), ktorý už nie je možné rozšíriť o žiadny susedný stĺpec.

Vašou úlohou je zistiť, koľko ramien bude mať rieka po každom poklese hladiny.

Formát vstupu a výstupu

V prvom riadku vstupu je počet stĺpcov n, na ktoré je rozdelený prierez koryta rieky.

V druhom riadku vstupu je n medzerami oddelených celých čísel – počiatočné hĺbky vody v jednotlivých stĺpcoch, v poradí, v akom nasledujú popri šírke koryta. Hĺbka každého stĺpca je celé číslo od 1 po n, hĺbky jednotlivých stĺpcov však *nemusia byť rôzne*.

Na výstup vypíšte n riadkov. V i-tom z nich (číslujúc od 1) je jedno celé číslo označujúce počet ramien rieky po i klesnutiach hladiny.

Obmedzenia a hodnotenie

Táto úloha má 5 hodnotených vstupov. Za každý správne vyriešený vstup dostanete dva body.

Vo všetkých vstupoch platí, že $1 \leq n \leq 10^6$.

V prvých troch vstupoch navyše platí, že $n \leq 1000$. V prvom a štvrtom vstupe má každý stĺpec inú hĺbku, počiatočné hĺbky teda tvoria permutáciu čísel 1 až n.

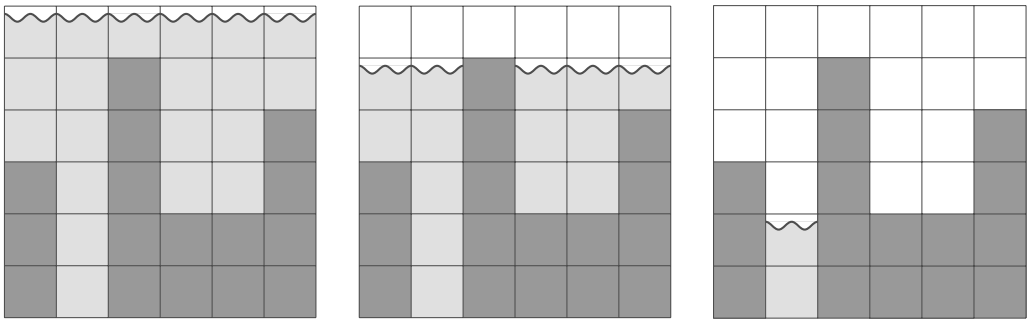
Príklad

vstup

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

výstup

```
2
2
2
1
1
0
```



Kým hladina neklesla, celú rieku tvorí jedno súvislé rameno. Len čo klesne o 1 jednotku, vynorí sa tretí stĺpec (hĺbka 1) a rieka sa rozdelí na dve ramená. Tieto dve ramená ostanú aj pri ďalších dvoch poklesoch, hoci sa bude meniť ich šírka. Po poklese o 4 jednotky zostáva pod vodou už len druhý stĺpec (hĺbka 6), rieka má teda jediné rameno. Po poklese o 6 jednotiek je koryto úplne suché, ramien rieky je teda nula.

Odovzdávanie riešení

Zo stránky `https://oi.sk/?d=rocnik` si stiahnite ZIP archív obsahujúci 5 testovacích vstupov.
Vstupy sú nazvané `01.txt` až `05.txt`.
Úlohu viete odovzdávať dvoma spôsobmi:

1. K čo najviac vstupom vytvoríte správne výstupy a uložíte ich do súborov `sol01.txt` až `sol05.txt`. Následne odovzdáte ZIP archív obsahujúci **zdrojový kód vášho programu** a tieto výstupné súbory.
2. Odovzdáte program v jednom z povolených programovacích jazykov (napr. `Python` alebo `C++`), ktorý číta štandardný vstup a vypisuje na štandardný výstup. Tento program bude automaticky otestovaný na tej istej sade vstupov, ktorá sa nachádza v ZIP archíve. Predlohu toho, ako má vyzerať takýto váš program, nájdete na stránke.

Za každý správny výstupný súbor získate 2 body.

B-I-3 Skladník Paľo

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

Paľo robí v sklade. Za roky práce sa mu tam nahromadilo veľké množstvo predmetov, ktoré už v podstate nikto nechce, a tak sa Paľo rozhodol trochu upratať. Pozoruhodné je, že všetky staré predmety majú rôznu hmotnosť, preto sa ich Paľo rozhodol usporiadať tak, aby mali jasnú štruktúru. Pomôžu mu pri tom extrémne pevné oceľové káble a gigantická jama, ktorú našiel v zadnej časti skladu.

Paľo pri upratovaní postupoval nasledovne. Na začiatku si zvolil jeden *základný* predmet, ktorý primontoval na okraj gigantickej jamy. K základnému predmetu priviazal dve oceľové laná, ktoré spustil do jamy – jedno viselo na ľavú stranu a druhé na pravú. Paľo potom zobral ďalší predmet a zavesil ho na koniec jedného z voľne visiacich lán. K novozavesenému predmetu opäť priviazal dve laná, jedno na ľavú, druhé na pravú stranu. Toto opakoval, kým nezavesil všetky predmety zo skladu.

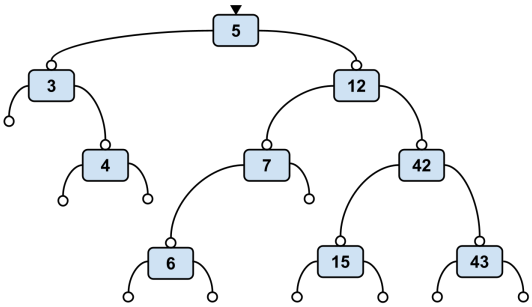
Vznikla mu tak veľká štruktúra zavesených predmetov. Kedže Paľo je vo voľnom čase horolezec, vie sa po nej pohybovať, vždy však **iba lezením po lanách**. Žiadne preskakovanie z predmetov na predmet či iné nebezpečné manévre. Ak stojí pri predmete P, môže buď zliezť dodola po ľavom alebo pravom lane, prípadne môže vystúpiť dohora po lane, na ktorom je predmet zavesený.

Povieme, že predmet Y je *zavesený naľavo* od predmetu X, ak sa vie Paľo presunúť od predmetu X k predmetu Y tak, že stále lezie iba dodola a **prvým lanom lezie doľava**. Zvyšné pohyby dodola už môžu byť po ľavom aj pravom lane. Rovnako, predmet Y je *zavesený napravo* od X, ak k nemu vie zliezť tak, že ako prvé použije pravé lano.

Aby sa vedel v štruktúre celý čas ľahko orientovať, chce aby celý čas platili nasledovné dve pravidlá:

1. Ku každému predmetu sú vždy priviazané práve tri laná – jedno doľava dole, jedno doprava dole a jedno, na ktorom je predmet zavesený. Výnimku tvorí základný predmet, ktorý nie je zavesený na žiadnom lane.
2. Pre **každý predmet** P platí, že predmety zavesené na jeho *ľavej strane sú ľahšie* ako P a predmety zavesené na *pravej strane sú ťažšie* ako P.

Na obrázku môžete vidieť jednu z možností, ako by mohla vyzerať Paľova štruktúra, ak by mal v sklade predmety s váhami 3, 4, 5, 6, 7, 12, 15, 42 a 43. Základný predmet je v tomto prípade predmet s váhou 5, ostatné predmety visia v jame.



Všimnite si, že naozaj platí, že predmety zavesené naľavo od každého predmetu sú ľahšie, a predmety zavesné napravo ťažšie ako daný predmet. Napríklad naľavo od predmetu 12 sú zavesené predmety 6 a 7, ktoré sú ľahšie, a napravo sú zavesené predmety 15, 42 a 43, ktoré sú ťažšie.

Špeciálne, predmet 6 je zavesený *napravo* od predmetu 5. Hoci na obrázku je 6 nakreslená fyzicky naľavo od 5, pri vzájomnej pozícii nás zaujíma len to, ktorým lanom musí začať Paľo zliezanie, čo je v tomto prípade pravé lano.

Takisto, predmet 6 nie je zavesený pod predmetom 3 (a to ani naľavo ani napravo), keďže z 3 sa k 6 Paľo nevie dostať iba zliezaním dodola.

Súťažná úloha

Úloha sa skladá z troch podúloh. V každej z nich navrhnite postup, ktorým sa má Paľo riadiť, aby vyriešil daný problém, a to bez ohľadu na to, aké predmety sa v jeho štruktúre nachádzajú a koľko ich je.

K riešeniu úlohy teda nestačí opísať správanie iba pre štruktúru na obrázku. Treba vymyslieť všeobecný postup, ktorý bude fungovať pre ľubovoľnú Paľovu štruktúru, a ktorý vie Paľo jednoducho nasledovať.

a) (3 body) Paľo stojí pri základnom predmete. Chcel by zistiť, či sa v štruktúre nachádza predmet s váhou w. Popíšte postup, pomocou ktorého sa Paľo na čo najmenej pohybov po lanách dostane k predmetu s váhou w, prípadne zistí, že sa daný predmet v štruktúre nenachádza.

b) (2 body) Paľovi odložili do skladu nový predmet s váhou w. Teraz stojí pri základnom predmete a chce predmet s váhou w umiestniť do štruktúry tak, aby stále platili všetky podmienky zo zadania.

Nechce však žiadne predmety premiestňovať, preto ho musí zavesiť na voľný koniec niektorého visiaceho lana. Navrhnite všeobecný postup, pomocou ktorého Paľo zavesí predmet s váhou w do štruktúry tak, aby neporušil žiadne z jej pravidiel.

Opäť sa snažte, aby bolo vaše riešenie čo najefektívnejšie, teda Paľo musel liezť cez čo najmenej lán.

c) (5 body) Z času na čas sa stane, že niektorý z Paľových predmetov si príde jeho vlastník vyzdvihnúť. Vtedy ho musí Paľo vybrať zo svojej štruktúry a prípadne štruktúru nejako preusporiadať, ak na vybranom predmete viseli ďalšie predmety.

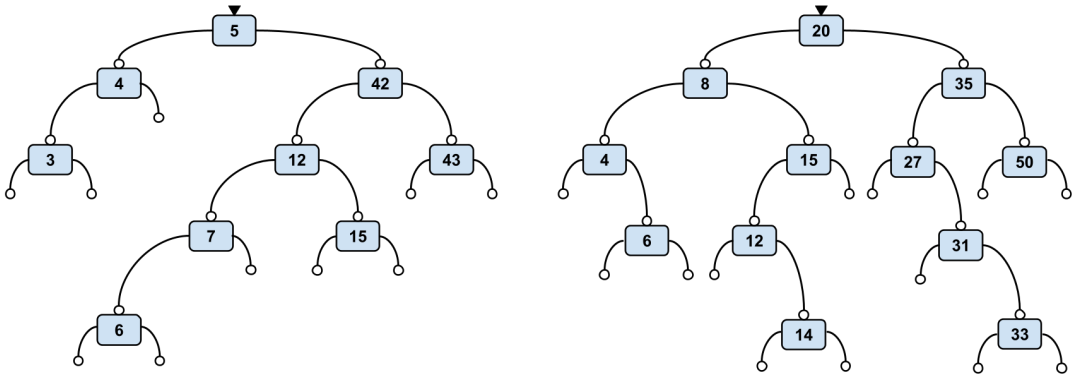
Paľo zliezol po lanách a teraz stojí pri predmete s váhou w, ktorý treba vrátiť jeho majiteľovi. Navrhnite postup, pomocou ktorého vie Paľo vybrať tento predmet a následne upraviť štrutkúru tak, aby opäť platili podmienky zo zadania.

V tomto riešení sa snažte o to, aby musel Paľo rozviazať a zaviazať čo najmenej lán. Nechce napríklad vybrať všetky predmety z jamy a nanovo ich tam všetky povkladať.

Príklady

Na obrázkoch nižšie sú uvedené ďalšie dva príklady, ako môže vyzerať Paľov sklad. Môžete ich použiť na otestovanie vašeho **všeobecného** postupu.

Všimnite si, že štruktúra naľavo obsahuje rovnakú množinu predmetov ako prvý obrázok v zadaní. Pre danú množinu totiž existuje viacero rôznych korektných usporiadaní, ktoré dodržujú všetky podmienky zadania.



B-I-4 Počítač so záhradnou hadicou

Toto je, netradične, ***praktická úloha****. V každej podúlohe odovzdaj príslušný program pre náš počítač.*

K tejto úlohe patrí študijný text uvedený na nasledujúcich stranách. Aj v nasledujúcom kole tohto ročníka OI stretneš úlohu nadväzujúcu na tento študijný text.

Ceny RAM pamäte idú stále hore. Futurologické oddelenie našej firmy predpovedá, že čochvíľa si digitálnu pamäť budú môcť dovoliť iba nadnárodné spoločnosti s kvadriliónmi dolárov. Po celom svete začalo pátranie po novom type pamäte, ktorý by si mohli kúpiť aj bežní smrteľníci. Zahraniční vedci skúšajú napríklad stádo slonov, obrovské kamenné tabule, zlaté gramofonové platne, alebo návrat k diernym štítkom. Naše hardvérové oddelenie zase zostrojilo počítač „Zahradník 8000“, ktorý si všetko ukladá do záhradnej hadice. Už len potrebujeme niekoho, kto nám preň všetko naprogramuje.

V domácom kole sme pre vás pripravili rozsiahlu sadu **menších podúloh**, aby ste sa so Zahradníkom 8000 mohli dobre oboznámiť. Svoje riešenia môžete testovať a odovzdávať cez samostatné webové rozhranie. Linku naň dostanete po prihlásení sa do systému na odovzdávanie riešení: https://oi.sk/oisubmit/

Podúlohy sú rozdelené do skupín. Každá skupina má určené maximum bodov, ktoré sa za ňu dajú získať. Keď túto hranicu dosiahnete, za vyriešenie ďalších podúloh v danej skupine už body nedostanete. Na získanie plného počtu bodov preto netreba vyriešiť všetky podúlohy.

Odporučame podúlohy riešiť v zadanom poradí, nie je to však nutné.

Študijný text: Zahradník 8000

Pamäť

Zahradník 8000 je náš nový počítač. Jeho hlavné dátové úložisko je záhradná hadica, v ktorej sú nasypané drevené guľôčky reprezentujúce čísla.

Hadica je jednosmerná – počítač vie čísla vhadzovať do jej horného konca a vďaka gravitácii ich **v tom istom poradí** vyberať z dolného konca. Guľôčky sa počas prechodu v hadici nevedia predbiehať ani vymieňať. Hadica je z gumy, takže sa do nej zmestí ľubovoľne veľa čísel. Počiatočný obsah hadice závisí od konkrétnej úlohy. Na záverečnom obsahu hadice nezáleží.

Keď niekde napíšeme, že hadica obsahuje $[10, 20, 30]$, myslí sa tým, že 10 je na spodku a vypadne ako prvé, potom 20, a nakoniec 30.

Okrem hadice má Zahradník 8000 ešte 26 premenných, ktoré voláme registre. Sú označené písmenami `a`...`z` a v každom môže byť jedno číslo. Na začiatku výpočtu je v každom registri nula.

Napokon má Zahradník 8000 ešte jeden kotúč bločkového termopapiera, na ktorý môže písať svoj výstup.

Všetky čísla, s ktorými Zahradník 8000 akokoľvek pracuje, sú **nezáporné** a správajú sa ako `uint64_t` v C/C++. Podúlohy sme zvolili tak, aby vás veľkosť čísel nemusela trápiť – bežné riešenia by sa do tohto rozsahu mali bez problémov zmestiť.

Príkazy

Programy pre Zahradníka 8000 sú textové súbory, ktoré obsahujú postupnosť príkazov. Pre stručnosť môžeme písať viac príkazov do jedného riadku, ale v takomto prípade ich od seba treba oddeliť bodkočiarkou. Prázdne riadky nič nerobia a nepočítajú sa ako príkaz. Komentáre sa začínajú znakom `#`. V nasledujúcej tabuľke sú zhrnuté všetky príkazy, ktoré Zahradník 8000 pozná, a ktoré môžete použiť pri programovaní.

príkaz	**účinok príkazu**
`get X`	Vyberie číslo z hadice a uloží ho do registra `X`.
`put X`	Vloží do hadice číslo z registra `X`.
`put číslo`	Vloží dané číslo do hadice.
`print`	Vyberie číslo z hadice a vypíše ho na výstup.
`add`	sčítanie: Vyberie dve čísla z hadice a vloží tam ich súčet.
`sub`	odčítanie: Vyberie dve čísla z hadice a vloží tam ich rozdiel (prvé mínus druhé).
`mul`	násobenie: Vyberie dve čísla z hadice a vloží tam ich súčin.
`div`	delenie: Vyberie dve čísla z hadice a vloží tam celú časť ich podielu (prvé lomeno druhé).
`mod`	zvyšok: Vyberie dve čísla z hadice a vloží tam zvyšok, ktorý dá prvé z nich po delení druhým.
`label L`	návestie: Toto miesto v programe dostane meno `L`. `L` môže byť ľubovoľný reťazec okrem medzier a znakov `#` a `;`
`jump L`	skok: Bude pokračovať vo vykonávaní programu od miesta, ktoré sa volá `L`.
`jz X L`	skok ak nula: Ak je v registri `X` nula, vykoná príkaz `jump L`.
`jeq X Y L`	skok ak sa rovnajú: Ak je v registroch `X` a `Y` to isté číslo, vykoná príkaz `jump L`.
`jgt X Y L`	skok ak je väčšie: Ak je v registri `X` väčšia hodnota ako v `Y`, vykoná príkaz `jump L`.
`jempty L`	skok ak je hadica prázdna: Ak v hadici nie sú žiadne čísla, vykoná príkaz `jump L`.
`stop`	koniec: Prestane vykonávať program.

Môžeme si všimnúť, že Zahradník 8000 nepozná príkazy ako `if` a `for`. Na implementáciu vetvenia a cyklov musíme preto použiť príkazy, ktoré skočia na iné miesto v programe.

Taktiež si všimnite, že aritmetická jednotka je bohužiaľ pripojená iba na hadicu, nie na registre. Všetky aritmetické príkazy vyberú dve čísla z hadice a vložia do nej výsledok. Podobne aj tlačová jednotka píše na výstup iba čísla z hadice. Ale naopak, porovnávacia jednotka dokáže čítať iba z registrov, nie z hadice.

Ak sa stane, že pri pokuse vybrať z hadice je hadica prázdna, nastane chyba. Rovnako nastane chyba, ak sa pokúsime deliť nulou, počítať zvyšok po delení nulou, alebo skočiť na neexistujúce miesto. Ak sa program dostane na koniec, Zahradník 8000 korektne skončí (ako keby na konci programu bol ešte príkaz `stop`).

Príklad: V hadici je jedno číslo, označme si ho n. Ak je $n > 100$, vypíšeme ho. Inak vypíšeme 0.

```
get n
put 100; get s    # Hadica je prázdna, môžeme do nej hodiť číslo 100 a hneď ho vybrať do registra s.
jgt n s vacsie
put 0             # <-- Ak n<=100, výpočet pokračuje ďalším riadkom.
print
stop

label vacsie      # <-- Ak n>100, výpočet skočí sem.
put n
print
```

ŠTYRIDSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Ján Gottweis, Jakub Prítrský, Jakub Šiagi, Tomáš Belan
Recenzia: Michal Anderle
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2026