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 2016**.
Riešenia kategórie B je potrebné odovzdať najneskôr **1. decembra 2016**.

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 Bežecké preteky

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

Olympijské hry v Riu boli plné úžasných športových výkonov v tých najrozličnejších disciplínach. Samkovi sa však najviac zapáčil šprint na 100 metrov. Páčil sa mu až tak, že ho neprestal sledovať ani po skončení olympiády a dokonca naň začal aj stávkovať.
Samko však vie, že hazard je nebezpečný a naviac, on sám má viac smoly ako šťastia. Stávkuje preto zásadne „na istotu“. Napríklad je v poriadku staviť na to, že Usain Bolt porazí Justina Gatlina, pretože ho predbehol v oboch posledných šprintoch, ktoré spolu bežali. Staviť ale na to, že Gatlin predbehne Yohana Blaka je nebezpečné. Gatlin ho síce porazil v poslednom spoločnom súboji, v behu pred tým však dominoval Blake.
Samka by teraz zaujímalo, koľko stávok na istotu vie spraviť, aby si vedel vypočítať, koľko na tom zarobí.

Súťažná úloha

Máme k dispozícii výsledky posledných dvoch šprintov. Každého šprintu sa zúčastnilo tých istých n bežcov. Zistite, koľko existuje takých dvojíc bežcov (a, b), že v oboch šprintoch skončil súťažiaci a pred súťažiacim b.

Formát vstupu

V prvom riadku vstupu sa nachádza kladné celé číslo n – počet bežcov, ktorý sa zúčastňujú súťaže. Bežcov si očíslujeme od 1 po n.
Druhý a tretí riadok obsahujú výsledky posledných dvoch šprintov týchto súťažiacich. Každý z týchto riadkov obsahuje n čísel – poradie, v ktorom títo súťažiaci dobehli do cieľa. (Teda na začiatku riadku je víťaz šprintu, za ním ten kto dobehol druhý, a tak ďalej.) V každom z týchto riadkov je teda nejaká permutácia čísel 1 až n.

Formát výstupu

Vypíšte jediné číslo – počet dvojíc (a, b) takých, že šprintér a skončil v oboch behoch pred šprintérom b.
Dajte si pozor, lebo výsledok sa **nemusí** zmestiť do 32-bitovej celočíselnej premennej. Použite preto premennú s dostatočným rozsahom – napr. `long long` v C++ alebo `int64` v Pascale.

Obmedzenia a hodnotenie

Je 10 sád testovacích vstupov, za každú môžete získať 1 bod. Vo všetkých sadách platí $n \leq 100\,000$.
V prvých 6 sadách naviac platí, že $n \leq 5\,000$. V prvých 3 sadách dokonca platí, že $n \leq 100$.
Časový limit je 0.5 sekundy. Pamäťový limit je 1 GB.

Príklady

vstup

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

výstup

```
10
```

Ľubovoľná dvojica, v ktorej je $a < b$, je správna. Takýchto dvojíc je práve 10.

vstup

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

výstup

```
5
```

Podmienky zo zadania spĺňajú dvojice $(2, 3)$, $(2, 4)$, $(5, 1)$, $(5, 3)$ a $(5, 4)$. Žiadna iná dvojica túto podmienku nespĺňa.

vstup

```
7
1 2 3 4 5 6 7
7 6 5 4 3 2 1
```

výstup

```
0
```

A-I-2 Rekonštrukcia školy

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

Opäť prišiel september a s ním aj školský rok. Pán učiteľ sa práve chystá ísť z kabinetu na svoju prvú hodinu. V budove školy sa však ešte dokončujú letné rekonštrukcie, a tak sa po chodbách povaľuje kopa harabúrd. No a náš pán učiteľ nie je práve najštíhlejší, takže sa mu nedarí dostať sa pomedzi všetky prekážky do triedy, kde má učiť. Možno keby šikovne kopol tuto do tej kopy starých vysvedčení, to by mu mohlo uvoľniť cestu...

Súťažná úloha

Pôdorys školy je obdĺžnik. Rozdelíme si ho na mriežku tvorenú $r \times s$ rovnako veľkými štvorcovými políčkami. Každé políčko je buď voľné, alebo je na ňom prekážka. Pán učiteľ v ľubovoľnom okamihu zaberá nejaký štvorec rozmerov 2×2. Učiteľ sa pohybuje v krokoch. V každom kroku sa posunie o jedno políčko (teda o polovicu svojej veľkosti) v jednom zo štyroch základných smerov.
Vašou úlohou je napísať program, ktorý načíta popis školy a nájde cestu medzi kabinetom (danou štvoricou voľných políčok) a triedou (inou danou štvoricou voľných políčok). Počas cesty sa pán učiteľ smie, až na nanajvýš jednu výnimku, pohybovať len po voľných políčkach. Presnejšie, trasa, po ktorej prejde pán učiteľ, smie obsahovať nanajvýš jedno políčko 1×1 na ktorom je prekážka. Všetky ostatné políčka, cez ktoré učiteľ prechádzal, musia byť voľné. (Na to jediné políčko s prekážkou smie učiteľ počas svojej cesty stúpiť ľubovoľne veľakrát, dokonca z neho môže odísť a neskôr sa naň zase vrátiť, ak chce.)

Formát vstupu

Prvý riadok vstupu obsahuje dve medzerou oddelené celé čísla r a s: počet riadkov a počet stĺpcov v mriežke. Platí $2 \leq r \leq 2000$ a $2 \leq s \leq 2000$. (Ako uvidíme nižšie, najmenší platný vstup má dokonca rozmery až 2×4.) Nasleduje mapa školy: r riadkov a v každom z nich s znakov. Bodky označujú voľné políčka, mriežky sú prekážky, písmená `K` a `T` označujú kabinet (kde učiteľ začína) a triedu (kam sa chce dostať). Môžete predpokladať, že na celej mape sú práve štyri `K` a štyri `T`. Aj `K` aj `T` budú vždy tvoriť štvorec rozmerov 2×2.

Formát výstupu

Vypíšte jediný riadok. Ak neexistuje žiadna vyhovujúca cesta, vypíšte text „`neexistuje`“. V opačnom prípade vypíšte popis ľubovoľnej vyhovujúcej cesty: reťazec písmen `N`, `E`, `S`, `W`. Tieto predstavujú krok na sever, východ, juh a západ. (Krok na sever dostane učiteľa na mape o riadok vyššie, krok na východ o stĺpec doprava.)
Vypísaná cesta *nemusí byť najkratšia možná*, jej dĺžka však nesmie prekročiť hodnotu $r \cdot s$.

Obmedzenia a hodnotenie

Riešenie, ktoré efektivne vyrieši všetky vstupy, získa 10 bodov. Až 7 bodov viete získať za vstupy kde $(r \cdot s) \leq 10\,000$, z toho 3 body sú za vstupy kde $(r \cdot s) \leq 30$. Časový limit je 5 sekúnd. Pamäťový limit je 1 GB.

Príklady

vstup

```
3 7
KK.#.TT
KK.#.TT
...#...
```

výstup

```
neexistuje
```

Nech by učiteľ išiel akokoľvek, jeho trasa by určite obsahovala aspoň dve prekážky.

vstup

```
4 7
KK.#.TT
KK.##TT
#......
...#...
```

výstup

```
ESSEEEENN
```

Iným správnym riešením pre tento vstup je reťazec `ESSEEEENSNN`.

A-I-3 Moderné umenie

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

V Kultúrnom Stredisku Postmoderny otvárajú novú výstavu obrazov maliara Mária. Jeho obrazy sú vrcholom abstraktného umenia: každý obraz je len zmes farebných bodiek na obrovskom bielom plátne. Zamestnanci strediska netušili, kde má ktorý obraz „hore“, a tak ich všetky zavesili rovnobežne so súradnicovými osami. Mário im však neskôr vysvetlil, že každý jeho obraz má jednu špeciálnu bodku, ktorá sa pri správnom zavesení nachádza na obraze napravo od všetkých zvyšných bodiek. Mário preto žiada pracovníkov KSP, aby obrazy pootáčali do správnych polôh. A oni žiadajú vás, aby ste im povedali, o koľko stupňov majú ktorý obraz otočiť.

Súťažná úloha

Plátno obrazu je štvorec, ktorého ľavý dolný roh sa nachádza na pozícii $(-10\,000, -10\,000)$ a pravý horný roh na pozícii $(10\,000, 10\,000)$. Otáčať ho môžete iba okolo jeho stredu, teda okolo pozície $(0, 0)$.
Na vstupe dostanete pozície jednotlivých farebných bodiek, pričom jedna z týchto bodiek je špeciálna. Vašou úlohou je nájsť také reálne číslo φ, že keď otočíme obraz okolo stredu o φ stupňov v smere hodinových ručičiek, tak bude mať špeciálna bodka ostro väčšiu x-ovú súradnicu ako hociktorá iná bodka.
Môžete predpokladať, že bodky sú navzájom rôzne a že žiadne tri bodky neležia na priamke. Naviac môžete pri písaní svojho programu predpokladať, že práca s reálnymi číslami je presná – teda nemusíte riešiť zaokrúhľovanie.

Formát vstupu a výstupu

V prvom riadku vstupu je číslo $n \geq 3$, udávajúce počet bodiek na obraze. Nasleduje n riadkov, každý obsahuje dve reálne čísla x_i a y_i $(-10\,000 \leq x_i, y_i \leq 10\,000)$ – súradnice jednej bodky. Špeciálna bodka je na vstupe uvedená ako prvá (t.j., jej súradnice sú v druhom riadku vstupu, hneď pod číslom n).

Na výstup vypíšte reálne číslo φ spĺňajúce podmienku zo zadania. Ak existuje viac riešení, vypíšte ľubovoľné z nich. Ak žiadne také číslo neexistuje, vypíšte `neexistuje`.

Hodnotenie

Za riešenie efektívne fungujúce pre $n \leq 100$ môžete získať až 5 body.
Za riešenie efektívne fungujúce pre $n \leq 1\,000$ môžete získať až 7 bodov.
Vzorové, 10-bodové riešenie by na bežnom počítači do sekundy vyriešilo ľubovoľný vstup s $n \leq 100\,000$.
Dôležitou súčasťou ľubovoľného efektívneho riešenia je **dôkaz jeho správnosti**.

Príklady

vstup

```
4
0 1
1 0
-1 0
0 -1
```

výstup

```
90.000000000
```

Špeciálny bod sa nachádza na pozícii $(0, 1)$. Ak celý obraz otočíme doprava o 90 stupňov, špeciálny bod skončí na súradniciach $(1, 0)$, zatiaľ čo ostatné body sa presunú na $(0, -1)$, $(0, 1)$ a $(-1, 0)$. Špeciálny bod má zjavne zo všetkých bodov najväčšiu x-ovú súradnicu. Iným správnym riešením by bolo napríklad číslo 87.75.

vstup

```
4
0 0
0 1
-1 -1
1 -1
```

výstup

```
Neexistuje
```

Nech obraz otočíme akokoľvek, špeciálny bod ostane stále na pozícii $(0, 0)$, lenže pri ľubovoľnom otočení bude niektorý zo zvyšných bodov viac napravo.

A-I-4 Stromochod

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 spolu nesúvisia, môžete ich riešiť v ľubovoľnom poradí. V každej podúlohe napíšte program pre stromochod tak, aby platilo: Keď ho spustíme na ľubovoľnom binárnom strome, tak v jeho koreni nastaví značku `ok` práve vtedy, ak ten strom spĺňa podmienku zadanú v príslušnej podúlohe.

Súťažná úloha, podúloha A (2 body)

Značku `ok` nastavte práve vtedy, ak je počet vrcholov v strome párny.

Súťažná úloha, podúloha B (4 body)

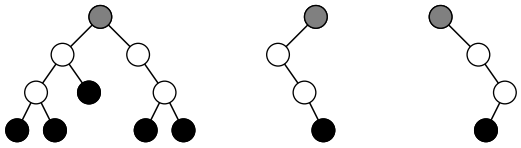
Značku `ok` nastavte práve vtedy, ak je počet vrcholov v strome mocninou dvojky (1, 2, 4, 8, ...).

Súťažná úloha, podúloha C (4 body)

Značku `ok` nastavte práve vtedy, ak je dotyčný strom úplný.
Strom je úplný, ak platia nasledovné podmienky:
- Všetky listy ležia rovnako hlboko. (T.j. do každého listu vedie z koreňa rovnako dlhá cesta.)
- Každý vrchol, ktorý nie je list, má oboch synov – aj ľavého, aj pravého.

Študijný text: stromochod

V tomto ročníku OI budeme programovať **stromochod** – zariadenie určené na výpočty na binárnych stromoch. Študijný text začneme definíciou toho, čo presne sú binárne stromy.
Graf sa skladá z konečnej množiny **vrcholov** a konečnej množiny **hrán**. Každá hrana spája dva vrcholy. Každé dva vrcholy sú spojené nanajvýš jednou hranou. **Cesta** je postupnosť *navzájom rôznych* vrcholov, v ktorej sú každé dva po sebe idúce vrcholy spojené hranou.
Strom je graf, v ktorom medzi každými dvoma vrcholmi vedie práve jedna cesta. Strom môžeme **zakoreniť** – jeden z jeho vrcholov prehlásiť za **koreň.** Pre každé dva vrcholy spojené hranou následne hovoríme, že ten z nich, čo je bližšie ku koreňu, je **otec** a ten druhý je jeho **syn**. Koreň je jediný vrchol, ktorý nemá otca. Vrcholy, ktoré nemajú žiadnych synov, nazývame **listy** stromu. **Podstrom** je časť stromu, ktorá je vytvorená daným vrcholom, jeho synmi, synmi jeho synov, a tak ďalej až k listom.
Binárny strom je špeciálny zakorenený strom. Každý vrchol v binárnom strome má najviac dvoch synov. Navyše (oproti všeobecnému binárnemu stromu) rozlišujeme **ľavých** a **pravých** synov – každý vrchol má teda nanajvýš jedného ľavého a nanajvýš jedného pravého syna. (Aj ak má vrchol len jedného syna, rozlišujeme, či je ľavý alebo pravý.)



Na obrázku vpravo vidíte niekoľko rôznych binárnych stromov. Stromy kreslíme tak, aby koreň bol úplne na vrchu obrázku. Na každom obrázku je koreň vyplnený sivou a všetky listy sú vyplnené čiernou farbou.
Stromochod slúži na riešenie problémov týkajúcich sa binárnych stromov. Môžeme si ho predstaviť ako počítač, ktorý sa pohybuje po strome. V každom okamihu výpočtu se nachádza v práve jednom z vrcholov stromu.
Pamäť stromochodu je ohraničená: môže si pamätať len konečné množstvo bitov informácie, teda v každom okamihu sa môže nachádzať len v jednom z konečného počtu stavov. Okrem toho stromochod smie zaznamenávať informácie len do navštívených vrcholov. Do každého vrchola sa taktiež zmestí len konečné množstvo informácie a tieto informácie môže stromochod čítať a zapisovať vždy len v tom vrchole, v ktorom sa práve nachádza. Údaje uložené vo vrchole budeme nazývať **značky.**
Stromochod môžeme programovať v ľubovoľnom bežnom programovacom jazyku. Kvôli ohraničeniu na konečný počet stavov môžeme používať len celočíselné premenné s vopred daným rozsahom (napríklad 0..9) a booleovské

premenné pre pravdivostné hodnoty. To napríklad znamená, že nie je možné používať polia ani smerníky. Z príkazov sú k dispozícii podmienky, cykly a goto. Procedúry a funkcie je možné používať, ale na ich parametre sa vzťahujú rovnaké ohraničenia na typy a navyše je zakázaná rekurzia. Pozor, použité rozsahy premenných **nesmú závisieť** od veľkosti stromu.
K aktuálnemu vrcholu (teda k tomu, v ktorom sa práve nachádza) vie stromochod pristupovať pomocou špeciálnej premennej „`V`“. Tá sa správa ako záznam (napr. `record` v Pascale, `struct` v C) ktorý obsahuje značky uložené v aktuálnom vrchole. Množinu značiek si môžete určiť vy pri písaní programu, ale musí ich byť konštantný počet a opäť platí ohraničenie na povolené typy.
Ak chceme stromochod presunúť na ľavého syna, pravého syna, resp. otca aktuálneho vrcholu, zavoláme funkciu `krok_l`, `krok_p`, resp. `krok_o`. Táto funkcia nemá žiadne parametre ani návratovú hodnotu. Ak sa pokúsite presunúť do neexistujúceho vrcholu, program sa zastaví s chybou. Preto sa hodí vopred otestovať, či syn alebo otec aktuálneho vrcholu existuje. Na to slúžia funkcie `ex_l`, `ex_p` a `ex_o`. Tieto pomocné funkcie nemajú žiadne parametre a vracajú booleovskú hodnotu.

Výpočet stromochodu prebieha následovne. Na vstupe dostaneme nejaký strom. Stromochod umiestnime **do jeho koreňa** a všetky značky v celom strome **vynulujeme** (teda číselné premenné nastavíme na nulu a boolovské na false). Výnimkou môžu byť značky, o ktorých je výslovne povedané, že sú súčasťou vstupu. Potom sa rozbehne program. Počas behu programu sa stromochod prechádza po strome, mení svoj stav (svoje lokálne premenné) a údaje zapísané vo vrcholoch stromu (značky). Keď beh programu korektne skončí, jeho výstup si prečítame z určených značiek.

Za čas behu programu na konkrétnom strome budeme považovať počet presunov po ňom. Časová zložitosť programu je funkcia, ktorá pre n vráti maximálny čas behu dotyčného programu na n-vrcholovom strome. Pamäťovú zložitosť neposudzujeme, pretože všetky programy si pamätajú lineárne množstvo informácie.

Príklad

Ukážeme si, ako stromochod naprogramovať tak, aby navštívil všetky vrcholy stromu a do každého umiestnil značku. Strom budeme prechádzať do hĺbky: začneme v koreni, odtiaľ prejdeme do ľavého podstromu. Keď sa z neho vrátime, presunieme sa do pravého podstromu, a nakoniec sa vrátime z celého podstromu von. (Na obrázku stromu takýto prechod zodpovedá prechádzaniu vrcholmi „proti smeru hodinových ručičiek“.)
Pretože nemáme k dispozíci rekurziu, budeme si v každom vrchole pamätať značku s menom `stav`, ktorá bude vyjadrovať, koľkokrát sme už vo vrchole boli. Ak navštívime vrchol prvýkrát, prejdeme na ľavého syna; druhýkrát prejdeme na pravého a po treťom príchode do vrcholu sa už vrátime na jeho otca. Ak ľavý alebo pravý syn neexistujú, budeme se chovať tak, ako by sme sa z nich vrátili okamžite. Program môže vyzerať takto:

```
var zbytocna: 0..47;      { ukazka lokalnej premennej stromochodu }

type vrchol = record      { typ "vrchol" popisuje, ake znacky ukladame do vrcholov }
   bol_som_tu: boolean;   { tuto znacku chceme nastavit na true vo vsetkych vrcholoch }
   stav: 0..3;            { pocitadlo kolkokrat sme uz tento vrchol navstivili }
end                       { na zaciatku je v kazdom vrchole bol_som_tu = false a stav = 0 };

begin
   zbytocna := 0;
   while zbytocna < 7 do begin { nekonecny cyklus }
      V.bol_som_tu := true;
      V.stav := V.stav + 1;
      case V.stav of
         1: if ex_l then krok_l;
         2: if ex_p then krok_p;
         3: if not ex_o then halt else krok_o;
      end;
   end;
end.
```

Časová zložitosť tohoto programu je lineárna od počtu vrcholov, pretože každý vrchol navštívime najviac trikrát.

TRIDSIATY DRUHÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Anderle, Michal Forišek
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2016