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 2013.

Riešenia kategórie A je potrebné odovzdať najneskôr **15. novembra 2013**.
Riešenia kategórie B je potrebné odovzdať najneskôr **30. novembra 2013**.

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 jazyku Pascal, C, alebo C++, 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.

Kategóriu B môžu riešiť len žiaci, ktorí ani v tomto, ani v nasledujúcom školskom roku nematurujú.
Jej riešenia je potrebné odovzdať na stránke `http://oi.sk/` najneskôr **30. novembra 2013**.

B-I-1 Klince v doske

Kleofáš má dlhú hrubú latu. A do tej laty má (do rôznej hĺbky) zatlčených n klincov, očíslovaných od 1 po n. Dĺžku tej časti klinca i, ktorá ešte trčí von z laty, označíme h_i.
Kleofáš má kladivo. Nemá ale ani kliešte, ani žiadne ďalšie klince. Preto jediné, čo teraz vie robiť, je zatĺcť nejaké klince *hlbšie* do laty (zmenšiť ich h_i). Kleofáš je majster v narábaní s kladivom: keď si povie ľubovoľnú novú hodnotu h_i (menšiu ako súčasná), vie ju vždy jediným úderom kladiva presne dosiahnuť.
Onedlho príde Kleofáša navštíviť Marienka. Kleofáš by jej chcel ukázať latu s klincami. Tá by bola najkrajšia vtedy, ak by všetky klince z nej trčali presne rovnako. Na to ale nie je čas, Marienka už je takmer za dverami!

Súťažná úloha

Na vstupe dostanete číslo n, čislo u a postupnosť $h_1, \ldots, h_n$ výšiek klincov. Číslo u udáva, že Kleofáš už má čas len na nanajvýš u úderov kladivom. Nájdite najväčšie m také, že je možné, aby Kleofáš pobúchal po klincoch tak, že m z nich bude z laty trčať rovnako. Tiež nájdite (jednu možnú) spoločnú výšku h ktorú bude mať po úprave laty týchto m klincov. (Môže byť aj $h = 0$. Klince zarovnané na výšku h nemusia susediť.)

Formát vstupu a výstupu

Dostanete od nás 5 vstupných súborov, označených `1.txt` až `5.txt`. Každý z nich obsahuje 6 testovacích vstupov. Každý testovací vstup je tvorený dvomi riadkami. V prvom riadku sú čísla n a k, pričom $0 \leq k \leq n$. V druhom riadku sú čísla $h_1, \ldots, h_n$, pričom $0 \leq h_i \leq 10^9$.
V jednotlivých vstupných súboroch platí $n \leq 10$, $n \leq 1000$, $n \leq 20\,000$, $n \leq 250\,000$ a $n \leq 1\,000\,000$.
V súboroch číslo 1, 2 a 4 navyše platí, že všetky výšky h_i sú z rozsahu $0 \leq h_i \leq 10^6$.

Pre každý testovací vstup vypíšte do príslušného výstupného súboru jeden riadok s dvomi číslami: optimálnym počtom zarovnaných klincov m a ich optimálnou výslednou výškou h. (Ak existuje viac možností pre h, vypíšte ľubovoľnú celočíselnú.) Správny výstupný súbor má teda obsahovať presne 6 riadkov.

Príklad

vstup

```
4 2
23 20 10 47
7 1
10 9 10 9 10 9 10
... (4 ďalšie vstupy) ...
```

výstup

```
3 10
4 10
... (4 ďalšie výstupy) ...
```

(V prvom príklade môže Kleofáš napr. klinec 1 a klinec 4 zarovnať na výšku 10. V druhom príklade je optimálne nič nerobiť. Iný správny výstup pre druhý príklad by bol „4 9“. Ten zodpovedá tomu, že Kleofáš jeden z klincov zníži z 10 na 9, čím dostane štyri klince trčiace po 9.)

Odovzdávanie riešení

Toto je praktická úloha. Napíšte **v ľubovoľnom programovacom jazyku** program, ktorý ju rieši.
Zo stránky `http://oi.sk/` stiahnite ZIP archív obsahujúci 5 testovacích vstupov, nazvaných `1.txt` až `5.txt`.
Vyrobte k čo najviac vstupom správne výstupy a uložte ich do súborov `sol1.txt` až `sol5.txt`.
Odovzdajte ZIP archív obsahujúci **zdrojový kód vášho programu** a tieto výstupné súbory.
Za každý správny výstupný súbor získate 2 body.

B-I-2 Prvočísla z magnetiek

Malá Paulínka má chladničku. Presnejšie, Paulínkina mama má chladničku. Paulínka má magnetky, ktoré na tú chladničku rada lepí. Na každej z magnetiek je jedna cifra (od 0 po 9).
Donedávna si Paulínka z cifier skladala čísla len tak, ako ju napadlo. No nedávno sa dozvedela o tom, že existujú prvočísla. Rada by si nejaké na chladničke zložila. Ešte však nevie, ako spoznať, či je nejaké číslo prvočíslo. Pomôžte jej!

Súťažná úloha

My vám povieme, aké cifry má Paulínka na magnetkách. Vašou úlohou je nájsť *všetky možné* prvočísla, ktoré sa dajú zložiť, ak použijeme *úplne všetky* cifry. (Cifru 0 nie je dovolené použiť na začiatku čísla.)

Prvočíslo je prirodzené číslo, ktoré má medzi prirodzenými číslami práve dva rôzne delitele. Najmenšie prvočísla sú teda 2, 3, 5, 7, 11, 13, atď.

Formát vstupu a výstupu

Zaujíma nás 5 rôznych sád cifier:

- 3 4 5 7
- 1 1 3 4 7
- 0 1 2 7 7 8 9
- 0 1 2 2 3 5 7 8 9
- 1 2 2 4 4 4 4 7 7 7 7

Každú sadu cifier vám dáme aj v súbore. Ten má v prvom riadku počet cifier a v druhom riadku ich zoznam.

Za správne riešenie pre každú sadu cifier sú 2 body. Správnym riešením je súbor, ktorý obsahuje zoznam čísel, ktorý má nasledovné vlastnosti:

- každé číslo v zozname je prvočíslom ktoré sa dá vyrobiť z daných cifier
- každé prvočíslo, ktoré sa dá vyrobiť z daných cifier, je v zozname práve raz
- zoznam je usporiadaný v rastúcom poradí

Príklad

vstup

```
4
0 1 1 8
```

výstup

```
1801
8011
8101
```

(Všimnite si, že každé z čísel 1801, 8011 a 8101 je vo výstupe len raz a že čísla 0181 a 0811 vo výstupe nie sú.)

Dobrá rada

Čísla, ktoré sa dajú vyrobiť z piatej sady cifier, sú už pomerne veľké. V niektorých programovacích jazykoch si treba dať pozor na to, aby sa vám zmestili do číselnej premennej. Napr. v jazyku C++ preto odporúčame používať typ `long long`, v Jave typ `long` a vo FreePascale typ `int64`.

Odovzdávanie riešení

Toto je praktická úloha. Napíšte **v ľubovoľnom programovacom jazyku** program, ktorý ju rieši.
Zo stránky `http://oi.sk/` stiahnite ZIP archív obsahujúci 5 testovacích vstupov, nazvaných `1.txt` až `5.txt`.
Vyrobte k čo najviac vstupom správne výstupy a uložte ich do súborov `sol1.txt` až `sol5.txt`.
Odovzdajte ZIP archív obsahujúci **zdrojový kód vášho programu** a tieto výstupné súbory.
Za každý správny výstupný súbor získate 2 body.

B-I-3 Pády domín

Janko raz ležal v nemocnici a strašne sa nudil. Keď to videla Peťka, doniesla mu sáčok plný rôznych domín. Janko sa veľmi potešil a hneď ich začal ukladať do rady a potom do nich strkal aby popadali. Domináma majú však rôzne váhy, preto nie vždy popadajú všetky. Presnejšie, i-te domino v rade má váhu v_i.
Janko môže strčiť ľubovoľné domino buď doprava alebo doľava. Ak strčí domino s váhou v, domináma padajú, až kým nenarazia na domino s váhou väčšou ako v. Vtedy sa pád zastaví.
Predstavme si, že Janko má rad domín s váhami: 2 4 3 8 1 3 9. Ak postrčí štvrté domino (s váhou 8) doprava, toto domino zhodí dve domináma – 1, 3 a zastaví sa pri narazení do 9. Ak strčí štvrté domino doľava, zhodí tri domináma, kým sa nedostane na koniec radu, kde sa pád zastaví.
Samozrejme sa Jankovi táto deštrukcia veľmi páči a páči sa mu tým viac, čím viac domín pri tom popadá. Pomôžte Jankovi zistiť, do ktorej strany má strčiť i-te domino, aby popadalo čo najviac domín.

Súťažná úloha

Dostanete popis radu domín, ktorý Janko postavil – pre každé domino jeho hmotnosť. Pre každé i zistite: Ak strčím domino i ako prvé, do ktorej strany ho mám strčiť, aby popadalo viac domín.

Formát vstupu a výstupu

Na vstupe dostanete popis radu domín. Na prvom riadku bude číslo n – počet domín v rade. Na druhom riadku bude n čísiel v_i – váhy domín v rade.

Na výstup vypíšte jeden riadok, ktorý obsahuje n znakov: pre každé domino jeden. Ak je lepšie i-te domino strčiť doľava, má byť i-ty znak „<“. Ak je lepšie strčiť ho doprava, má byť i-ty znak „>“. A ak sú obe možnosti rovnako dobré, vypíšte „=“.

Hodnotenie

Za riešenie podobne efektívne ako vzorové môžete získať plných 10 bodov. Za ľubovoľné správne riešenie získate od 3 po 6 bodov, v závislosti od časovej zložitosti daného riešenia.

Príklad

vstup	výstup
7 2 4 3 8 1 3 9	===<=<<

Ak domino 4 (s váhou 8) zhodíme doľava, padnú dokopy 4 domináma. Ale ak ho zhodíme doprava, padnú len 3. Preto je 4. znak výstupu <.

Odovzdávanie riešení

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

B-I-4 Hľadáme v poli

Keď v počítači spracúvame dáta, často v nich potrebujeme vyhľadávať. V tejto úlohe sa budeme zaoberať jednou z najjednoduchších situácií, kedy v dátach vyhľadávame: Máme *neprázdne* pole A *navzájom rôznych* celých čísel, ktoré sú *usporiadané podľa veľkosti* v rastúcom poradí. A ďalej máme dané celé číslo x, o ktorom nás zaujíma, či sa v poli A nachádza, a ak áno, *na ktorom indexe leží*.
Túto úlohu sa samozrejme dá riešiť hrubou silou: Postupne porovnáme x s každým prvkom poľa A. Ak niekedy nastane rovnosť, vrátime na výstup zodpovedajúci index, ak nikdy rovnosť nenastane, podáme správu, že x sa v poli A nenachádza. Existujú však aj omnoho efektívnejšie spôsoby riešenia tejto úlohy.

Vy túto úlohu však riešiť nebudete. Pre vás sme si pripravili niečo úplne iné. Trom študentom sme zadali nasledovnú úlohu: „Naprogramujte čo najefektívnejšiu funkciu, ktorá na vstupe dostane pole A a číslo x spĺňajúce vyššie popísané podmienky a na výstupe vráti buď číslo -1, ak sa x v poli A nenachádza, alebo index i taký, že $A[i] = x$.“ Každý z našich troch študentov stvoril nejaké riešenie. No a vašou úlohou bude teraz tieto tri riešenia skontrolovať a zistiť, či fungujú.
Ak nejaké riešenie nefunguje, stačí nájsť jeden konkrétny protipríklad: teda jedno pole A a číslo x, pre ktoré naprogramovaná funkcia nespraví to, čo sme chceli. Pole A vo vašom protipríklade nesmie mať viac ako 16 prvkov a tieto prvky musia byť z rozsahu od 0 po 1 000 000. Ak nejaké riešenie funguje, zdôvodnite, prečo to tak je, a odhadnite jeho časovú zložitosť. Pozor, hodnotiť máte funkčnosť konkrétnej implementácie, nie len správnosť hlavnej myšlienky riešenia.

V nasledujúcom texte nájdete tri spomínané programy v jazyku Pascal. Na webe OI na adrese `http://oi.sk/archiv/2013/b14.html` si môžete v prípade potreby pozrieť ekvivalentné programy v C++ a v Pythone.

Andrej vám k svojmu programu zanechal nasledovný popis:

Pozriem sa na prvý a na posledný prvok. Z nich si spočítam, kde vychádza, že bude x. Napríklad ak je prvý prvok 100, posledný prvok 1000 a ja hľadám 400, tak bude asi niekde v tretine poľa.
Presnejšie to funguje takto: nech práve hľadám v poli na pozíciách p až q. Pozrieme sa na hodnoty $A[p]$ a $A[q]$. Teraz si predstavme, že všetky prvky na pozíciách p až q tvoria aritmetickú postupnosť. Aké by mali hodnoty? Označme $d = (A[q] - A[p])/(q - p)$. Potom tieto prvky majú hodnoty $A[p]$, $A[p] + d$, $A[p] + 2d$, a tak ďalej. A my teraz hľadáme nejaké x, hej? No tak si nájdeme také i, aby $A[p] + id$ bolo čo najbližšie ku x, a budeme ho hľadať tam.
Jasne že sa nemusím vždy na prvý krát trafiť presne. Ale aj keď sa netrafím presne, tak aspoň viem nejaké prvky zahodiť a ďalej hľadať v kratšom úseku poľa. A keď to celé pre istotu zopakujem 10-krát tak už to určite sadne.

Listing programu (Pascal)

```
{ hľadáme hodnotu X v poli A[0..N-1], o ktorom predpokladáme, že je usporiadané vzostupne }

function andrej_hlada(var A : array of longint; N : longint; X : longint) : longint;

var kolo, odpoved, prvy, posledny, kde : longint;

begin
    prvy := 0; posledny := N-1;
    odpoved := -2;
    for kolo := 1 to 10 do begin
        if (X < A[prvy]) then odpoved := -1;
        if (X = A[prvy]) then odpoved := prvy;
        if (X = A[posledny]) then odpoved := posledny;
        if (X > A[posledny]) then odpoved := -1;
        if (odpoved = -2) and (prvy + 1 >= posledny) then odpoved := -1;
        if (odpoved <> -2) then break;

        kde := prvy + round( (posledny - prvy) * (X - A[prvy]) / (A[posledny] - A[prvy]) );
        if (kde < prvy+1) then kde := prvy+1;
        if (kde > posledny-1) then kde := posledny-1;

        if (X < A[kde]) then posledny := kde;
        if (X = A[kde]) then begin odpoved := kde; break; end;
        if (X > A[kde]) then prvy := kde;
    end;
    if (odpoved = -2) then odpoved := -1;
    andrej_hlada := odpoved;
end;
```

Boris len naškrabal na kus papiera stručný odkaz: „To je ľahké! Použijem binárne vyhľadávanie. Kým mám viac možností, tak sa vždy pozriem do stredu a podľa toho viem, či hľadať x vľavo alebo vpravo. A keď mi už ostane len jedna možnosť, tak pozriem, či je to presne to čo hľadám alebo nie.“

Listing programu (Pascal)

```
function boris_hlada(var A : array of longint; N : longint; X : longint) : longint;

var prvy, posledny, stredny : longint;

begin
    prvy := 0;
    posledny := N-1;
    while prvy < posledny do begin
        stredny := (prvy + posledny) div 2;
        if (A[stredny] <= X) then prvy := stredny else posledny := stredny;
    end;
    if (A[prvy] = X) then boris_hlada := prvy else boris_hlada := -1;
end;
```

Cilka vám ku svojmu programu nenapísala vôbec nič.

Listing programu (Pascal)

```
function cilka_hlada(var A : array of longint; N : longint; X : longint) : longint;

var kde, krok : longint;

begin
    cilka_hlada := -1;
    if (X < A[0]) then exit;

    krok := 0;
    while krok*krok < N do inc(krok);
    kde := 0;
    while (kde+krok<N) and (A[kde+krok] <= X) do kde := kde+krok;
    krok := 1;
    while (kde+krok<N) and (A[kde+krok] <= X) do kde := kde+krok;
    if (A[kde] = X) then cilka_hlada := kde;
end;
```

Odovzdávanie riešení

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

DVADSIATY DEVIATY ROČNÍK OLYMPIÁDY V INFORMATIKE

Príprava úloh: Michal Forišek, Michal Anderle
Recenzia: Michal Forišek
Slovenská komisia Olympiády v informatike
Vydal: IUVENTA – Slovenský inštitút mládeže, Bratislava 2013