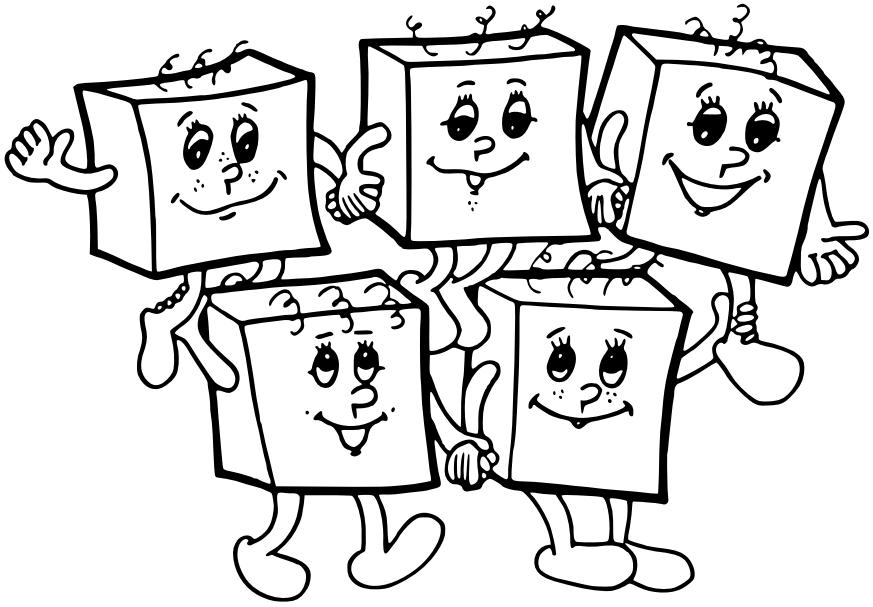
OLYMPIÁDA V INFORMATIKE
NA STREDNÝCH ŠKOLÁCH



dvadsiaty štvrtý ročník
školský rok 2008/09

zadania domáceho kola
kategórie A a B

- **Olympiáda v informatike** je od školského roku 2006/07 samostatnou súťažou. Predchádzajúcich 21 ročníkov tejto súťaže prebiehalo pod názvom **Matematická olympiáda, kategória P** (programovanie).
- Oficiálnu **webstránku** súťaže nájdete na `http://ksp.sk/oi/`.

Informácie a pravidlá

Pre koho je súťaž určená?

OI sa uskutočňuje v týchto kategóriách:

Kategória B má dve kolá: domáce a krajské.
Do kategórie B sa smú zapojiť len tí žiaci, ktorí ešte ani v tomto, ani v nasledujúcom školskom roku nebudú končiť strednú školu.
Kategória A má tri kolá: domáce, krajské a celoštátne.
Do kategórie A sa môžu zapojiť všetci žiaci (základných aj) stredných škôl.
Najlepší riešitelia kategórie A majú šancu reprezentovať Slovensko na medzinárodných súťažiach.

Zmena oproti minulému ročníku

V tomto ročníku olympiády prechádzame na nový spôsob odovzdávania riešení domáceho kola.

Dve z úloh domáceho kola sú zamerané prakticky. Vašou úlohou je vytvoriť a odladiť funkčný a čo najviac efektívny program (v jazyku Pascal, C, alebo C++), ktorý bude zadanú úlohu riešiť. **Riešenia týchto úloh budete odovzdávať prostredníctvom webového rozhrania**, ktoré nájdete na stránkach olympiády (`http://ksp.sk/oi/`) najneskôr v októbri 2008.

Odovzdaný program bude automaticky otestovaný na viacerých vopred pripravených vstupných 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.

Zvyšné dve úlohy sú teoretické. Ich riešenia môžete odovzdať **buď klasickou cestou** (odoslaním poštou na adresu príslušnej krajskej komisie), **alebo vo formáte PDF** prostredníctvom vyššie spomínaného webového rozhrania. Tieto úlohy budú opravené a vyhodnotené po uplynutí termínu na odovzdávanie úloh domáceho kola.

Priebeh súťaže

Za každú úlohu domáceho kola sa dá získať od 0 do 10 bodov. Na základe bodov domáceho kola stanoví Slovenská komisia OI (samostatne 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 A sú do **celoštátneho kola** pozývaní najúspešnejší riešitelia krajských kôl. Presnejšie, po vyhodnotení krajských kôl prebehne koordinácia bodovacích škál, spoja sa výsledkové listiny do jednej celoštátnej, a do celoštátneho kola sú pozvaní najlepší riešitelia podľa tejto výsledkovej listiny.

V celoštátnom kole účastníci prvý deň riešia tri teoretické úlohy, druhý deň dve praktické úlohy (pri počítači). Z najlepších riešiteľov tohto kola SK OI na výberovom sústredení vyberie družstvá pre Medzinárodnú informatickú olympiádu (IOI) a Stredoeurópsku informatickú olympiádu (CEOI).

Ako majú vyzerať riešenia úloh?

Presný popis, ako majú vyzerať riešenia praktických úloh, 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 napísaný v jazyku Pascal, C alebo C++.

Usporiadateľ súťaže

Olympiádu v informatike (OI) vyhlasuje *Ministerstvo školstva SR* v spolupráci so *Slovenskou informatickou spoločnosťou* 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.

Zadania kategórie B

Riešenia kategórie B treba odovzdať najneskôr **1. decembra 2008**.
V prípade riešení odoslaných poštou rozhoduje pečiatka z pošty.

B-I-1 Diaľnica

Píše sa rok 2108. Bratislavu a Košice konečne spojila diaľnica. Keď to Jano, bývajúci v Košiciach, uvidel, od radosti sa hneď rozbehol kúpiť si auto.

Rýchlo však zistil, že auto nejazdí na dobré slovo, ale treba doň občas natankovať benzín. Keď natankuje plnú nádrž, dokáže na svojom aute prejsť po diaľnici až K metrov.

Pozdĺž diaľnice sa nachádza N čerpacích staníc, príčom i-ta stanica sa nachádza a_i metrov od Košíc. Jana zaujímajú také dvojice staníc, ktorých vzdialenosť je menšia ako dojazd jeho auta – teda ak na jednej z nich natankuje, dokáže sa potom dostať na druhú z nich.

Súťažná úloha

Napíšte program, ktorý načíta hodnoty K, N a a_i a spočíta počet dvojíc čerpacích staníc, ktorých vzdialenosť je menšia alebo rovná K.

Formát vstupu

Prvý riadok vstupu obsahuje jedno celé číslo K ($1 \leq K \leq 1\,000\,000\,000$) – maximálnu vzdialenosť, ktorú prejde Janovo auto na jedno natankovanie.

Druhý riadok obsahuje počet čerpacích staníc N ($1 \leq N \leq 200\,000$, v polovici testovacích vstupov bude dokonca platiť $1 \leq N \leq 2\,000$).

Tretí riadok obsahuje N medzerami oddelených celých čísel $a_1, \ldots, a_N$ ($0 \leq a_1 < a_2 < \cdots < a_N \leq 1\,000\,000\,000$) – vzdialenosti jednotlivých čerpacích staníc od Košíc.

Formát výstupu

Výstup má obsahovať jediný riadok a v ňom jediné celé číslo P – počet dvojíc čerpacích staníc, ktorých vzdialenosť je najviac K.

Môžete predpokladať, že testovacie vstupy budú zvolené tak, aby hodnota P nikdy neprekročila 2 miliardy.

Príklady

vstup

```
100
3
15 80 120
```

výstup

```
2
```

Prvá a tretia stanica sú vo vzdialenosti 105, čo je priveľa.

vstup

```
100
5
15 65 80 100 115
```

výstup

```
10
```

Každé dve stanice sú vo vzdialenosti 100 alebo menej.

Odovzdávanie riešení

Toto je praktická úloha. Odovzdávate **len funkčný, odladený program** prostredníctvom webového rozhrania.

B-I-2 Kolotoč

V lunaparku majú nový kolotoč. Je unikátny tým, že ide o jediný štvorcový kolotoč na celom svete. Presnejšie, tvorí ho N^2 sedadiel umiestnených do štvorca veľkosti $N \times N$.

Keď je kolotoč pustený, každú sekundu sa sedadlá presunú na nové pozície. Jedno otočenie vyzerá nasledovne: Sedadlá, ktoré sú na obvode kolotoča, sa posunú o 1 v smere hodinových ručičiek. Keď si tieto sedadlá odmyslíme, dostaneme kolotoč $(N-2) \times (N-2)$, ktorý otočíme rovnakým spôsobom. (Pre nepárne N nám v strede ostane jedno sedadlo, ktoré sa točí na mieste.)

Súťažná úloha

Daná je hodnota N, mená detí sediacich na kolotoči, a kladné číslo K. Napíšte program, ktorý načíta tieto údaje a vypíše, ako bude kolotoč vyzerať po K sekundách točenia sa.

Formát vstupu

Prvý riadok vstupu obsahuje jedno celé číslo N ($1 \leq N \leq 1\,000$) – rozmer strany kolotoča.

Nasleduje N riadkov, každý obsahuje N mien detí. Mená detí sú tvorené veľkými a malými písmenami a majú dĺžku 1 až 6 znakov. Medzi každými dvoma menami je jedna medzera. Na kolotoči môže sedieť viac detí s rovnakými menami.

V poslednom riadku je jedno celé číslo K ($1 \leq K \leq 1\,000\,000$) – počet otočení kolotoča.

V polovici testovacích vstupov bude navyše platiť $N, K \leq 100$.

Formát výstupu

Výstup má obsahovať N riadkov, v ktorých bude popísaný záverečný stav kolotoča. Všetky mená vypíšte zarovnané doprava na 7 znakov. (Dĺžka každého riadku výstupu teda musí byť presne $7N$.) Kratšie mená doplňte zľava medzerami.

Rada: v jazyku C môžete na správne formátovaný výpis premennej `meno` použiť príkaz `printf("%7s",meno);` a v jazyku Pascal príkaz `write(meno:7);`

Príklad

vstup

```
4
Alenka Julka Monika Danka
Lucka Misko Mirko Petko
Jozko Milan Risko David
Jano Katka Misko Martin
2
```

výstup

```
  Jozko  Lucka Alenka  Julka
   Jano  Risko  Milan Monika
  Katka  Mirko  Misko  Danka
  Misko Martin  David  Petko
```

Odovzdávanie riešení

Toto je praktická úloha. Odovzdávate **len funkčný, odladený program** prostredníctvom webového rozhrania.

B-I-3 Balíčky

Mnohé distribúcie operačných systémov poskytujú svojim používateľom softvér rozdelený do balíčkov. Používateľ si prezrie zoznam ponúkaných balíčkov, vyberie si tie, ktoré chce používať, a operačný systém mu automaticky nainštaluje všetko potrebné.

Z pohľadu používateľa je to jednoduché a príjemné. O niečo ťažšie to má operačný systém. Niektoré balíčky totiž na sebe závisia. Presnejšie, môže sa stať, že na to, aby balíček i fungoval správne, treba mať nainštalované aj balíčky j a k.

V tejto úlohe sa budeme zaoberať zjednodušenou verziou takéhoto systému balíčkov.

Súťažná úloha

Používateľ má k dispozícii N rôznych balíčkov, ktoré ešte nemá nainštalované. Pre každý balíček i máme danú jeho veľkosť v_i a zoznam iných balíčkov, na ktorých tento balíček závisí. (Ak balíček i závisí na balíčkoch j a k, tak ak chceme nainštalovať balíček i, musíme nainštalovať aj j, aj k.)

Používateľ si vybral M balíčkov, ktoré potrebuje. Zistite celkovú veľkosť balíčkov, ktoré treba nainštalovať.

Formát vstupu

Prvý riadok vstupu obsahuje jedno celé číslo N ($1 \leq N \leq 10\,000$) – počet balíčkov. Balíčky majú priradené čísla od 1 po N.

Nasledujúcich N riadkov popisuje jednotlivé balíčky. Presnejšie, i-ty riadok začína názvom balíčka i a jeho veľkosťou v_i ($1 \leq v_i \leq 10\,000$) v kilobajtoch. Nasleduje počet p_i iných balíčkov, na ktorých balíček i závisí. Zvyšok riadku tvorí p_i navzájom rôznych čísel týchto balíčkov.

Môžete predpokladať, že $p_1 + \cdots + p_N \leq 100\,000$ a že v závislostiach medzi balíčkami nie sú cykly. (Nemôže sa teda stať že by napr. balíček i závisel na balíčku j, ten na balíčku k, a ten na balíčku i.)

Názvy balíčkov neobsahujú medzeru a majú najviac 20 znakov. Medzi každou dvojicou údajov v riadku je práve jedna medzera.

V predposlednom riadku vstupu je jedno celé číslo M ($0 \leq M \leq N$) – počet balíčkov, ktoré chce používateľ.

V poslednom riadku je M rôznych čísel (oddelených medzerami), popisujúcich tieto balíčky.

Formát výstupu

Výstup má obsahovať jediný riadok a v ňom celkovú veľkosť balíčkov v kilobajtoch, ktoré treba nainštalovať.

Príklad

vstup

```
6
python 300 1 6
bittorrent 500 2 1 3
ncurses 550 1 6
gdb 4700 0
freepascal 1200 2 4 6
glibc 100 0
1
2
```

výstup

```
1450
```

Používateľ chce balíček `bittorrent`. *Tento závisí na balíčkoch* `python` a `ncurses`, *ktoré oba závisia na balíčku* `glibc`.
Všimnite si, že balíček `glibc` *stačí nainštalovať raz. Ostatné dva balíčky nainštalovať netreba.*

Odovzdávanie riešení

Toto je teoretická úloha. Riešenie, spĺňajúce požiadavky uvedené v pravidlách, buď pošlite poštou na adresu vašej krajskej komisie, alebo odovzdajte vo formáte PDF prostredníctvom webového rozhrania.

B-I-4 Divné jazyky

Kdesi v oceáne medzi Polynéziou a Patagóniou objavili nedávno moreplavci časom zabudnuté súostrovie, ktoré nazvali Polygónia. Na mnohých ostrovoch tohto súostrovia žijú domorodé kmene, ktoré používajú vskutku podivné jazyky. Moreplavci zistili, že tieto kmene poznajú len dve hlásky: U a A. Z týchto hlások sú zložené všetky ich slová.

Súťažná úloha

a) (1 bod)
V kmeni Krivozubých používajú jazyk, ktorý má nasledujúce vlastnosti:
– Žiadne slovo nie je dlhšie ako 8 znakov.
– Každé slovo znie odpredu rovnako ako odzadu.

Teda napríklad môžu používať slová A, AA a $UAUAU$, ale nie $UUAA$ (lebo odzadu je to $AAUU$, čo nie je to isté ako $UUAA$) ani $AAAUUUAAA$ (lebo je pridlhé).

Koľko najviac slov môže mať ich jazyk?

b) (2 body)
Aj v kmeni Dlhonosých používajú jazyk, kde každé slovo znie odpredu rovnako ako odzadu. Avšak Dlhonosí majú väčšiu slovnú zásobu, ich slová môžu mať nanajvýš 47 znakov. Koľko najviac slov môže mať ich jazyk?

c) (3 body)
Kmeň Strapatých má naozaj podivuhodný jazyk. Všetky slová v ich jazyku majú presne 7 hlások. Navyše platí, že ak sa domorodec z tohto kmeňa pomýli v ľubovoľnej jednej hláske ľubovoľného slova, aj tak sa dá spoznať, ktoré slovo chcel povedať.

V jazyku tohto kmeňa teda napríklad nemôže byť dvojica slov $UUUAAAU$ a $UUUAAUA$ – keby niekto povedal $UUUAAAA$, nebolo by jasné, ktoré z týchto dvoch slov chcel povedať. Z rovnakého dôvodu môže byť v tomto jazyku najviac jedno zo slov $UUUAAAU$ a $UUUAAAA$.

Nájdite jazyk, ktorý spĺňa tieto požiadavky a má čo najviac slov. (Ak chcete, môžete si samozrejme pri hľadaní pomôcť počítačom, nezabudnite však v takom prípade v riešení uviesť aj použitý program.)

d) (4 body)
Dokážte, že jazyk kmeňa Strapatých nemôže mať viac ako 16 slov.

Odovzdávanie riešení

Toto je teoretická úloha. Riešenie, spĺňajúce požiadavky uvedené v pravidlách, buď pošlite poštou na adresu vašej krajskej komisie, alebo odovzdajte vo formáte PDF prostredníctvom webového rozhrania.

Zadania kategórie A

Riešenia kategórie A treba odovzdať najneskôr **17. novembra 2008**.
V prípade riešení odoslaných poštou rozhoduje pečiatka z pošty.

A-I-1 Meníme históriu

Pred veľa, veľa rokmi, keď sa ešte voda sypala a piesok lial, vládol v krajine Siedmeho sveta kráľ Ogrgeľ Veľký. I zaumienil si on, že by bolo treba zmeniť učebnice dejepisu. Tie totiž napísal kráľ Papľuh IV. A samozrejme v nich tvrdí, že on, Papľuh IV., je ten najväčší, najmúdrejší a najurodzenejší. Toto však nie je pravda – Ogrgeľ Veľký je predsa ten, ktorý je najväčší, najmúdrejší a najurodzenejší. Tú veľkosť má dokonca aj v mene! Všetky učebnice by bolo treba zmeniť... Lenže s tým je práca. A keď je on ten najmúdrejší, nebude ju predsa robiť on.

Súťažná úloha

Kráľ zostavil zoznam dvojíc slov. Každú dvojicu slov tvorí jedno staré nevhodné slovo, a jedno nové vhodné, ktorým treba to staré nahradiť. Napíšte program, ktorý načíta zoznam dvojíc slov a vstupný text, a následne vo vstupnom texte všetky výskyty nevhodných slov nahradí ich novými, vhodnými „ekvivalentmi“.

Formát vstupu

V prvom riadku súboru sa nachádza prirodzené číslo N ($N < 100\,000$), ktoré udáva počet nevhodných slov. Nasleduje N riadkov, pričom v každom sa nachádzajú vždy dve slova oddelené medzerou. Slová sú tvorené iba z veľkých písmen anglickej abecedy. Prvé slovo v každom z týchto N riadkov je nevhodné, druhé je jeho náhrada. Od riadku $N + 2$ až do konca súboru nasleduje samotný text učebnice. Text je tvorený iba z veľkých písmen anglickej abecedy a medzier (a koncov riadkov), pričom súvislé úseky písmen tvoria jednotlivé slová. Môžete predpokladať, že žiadné slovo nebude dlhšie ako 255 písmen a že žiadne dve nevhodné slová nebudú rovnaké.

Vstupné súbory použité pri testovaní budú mať unixové konce riadkov – každý riadok, vrátane posledného, končí znakom line feed (ASCII hodnota 10).

Formát výstupu

Výstupom programu je text, v ktorom boli všetky nevhodné slová nahradené ich vhodnejšími ekvivalentmi. Zvyšok súboru musí zostať bez zmeny, vrátane rovnakých medzier a rozdelenia slov do riadkov.

Príklad

vstup

```
5
PAPLUH OGRGEL
OGRGEL PAPLUH
DRACICI DRAKOVI
KRK REBRO
ZBYTOCNA DVOJICA
A TAK SA STALO ZE VELKY PAPLUH
KRKOLOMNE ZLOMIL DRACICI KRK
 V TEJTO UCEBNICI   BUDE HANENY OGRGEL

JANOSIKA ZAVESILI ZA PIATE REBRO
```

výstup

```
A TAK SA STALO ZE VELKY OGRGEL
KRKOLOMNE ZLOMIL DRAKOVI REBRO
 V TEJTO UCEBNICI   BUDE HANENY PAPLUH

JANOSIKA ZAVESILI ZA PIATE REBRO
```

Všimnite si, že slová `KRKOLOMNE` *(druhý riadok) ani* `REBRO` *(posledný riadok) sa nezmenili.*

Odovzdávanie riešení

Toto je praktická úloha. Odovzdávate **len funkčný, odladený program** prostredníctvom webového rozhrania.

A-I-2 Prechádzka po kanáloch

Mnoho rokov prebehlo, odkedy Jean Valjean utekal Parížskymi kanálmi, avšak ani napriek tomu nestratili nič zo svojej majestátnosti a tajomnosti. A dnes sa Charlotte a Jacques rozhodli tieto kanály preskúmať. Charlotte sa ale kanálov bojí, preto ostane sedieť vonku a vysielačkou bude rozprávať s Jacquesom. Ten bude kráčať kanálmi a hľadať artefakty a zaujímavosti (nazvime ich súhrnne *poklad*), ktoré Jean vo svojej mape vyznačil.

Podľa plánov, ktoré našli v tajnom denníku Jeana Valjeana, sú kanály pod Parížom orientované podľa svetových strán a zodpovedajú mape na štvorčekovom papieri. Každý štvorček na mape predstavuje políčko so stranou 1 m. V kanáloch občas bývajú mreže s dvierkami, ktoré sa dajú zamknúť. Každá mreža zaberá jedno políčko. Keď je zamknutá, nedá sa cez zodpovedajúce políčko ísť. Jean našťastie počas svojich potuliek kanálmi našiel a do mapy zakreslil všetky mreže. A čo je ešte dôležitejšie, získal aj kľúče k nim a poukrýval ich na rôznych miestach v kanáloch. Zámky na mrežiach aj kľúče sú štyroch farieb: červené, modré, zelené a fialové. Keď máme kľúč, vieme ním otvárať zámky jeho farby.

Aby sa Jacques nestratil, chodí krokmi dlhými 1 m a otáča sa len o násobky 90°, t.j. môže sa pohybovať len vodorovne alebo zvisle na mape. Jacques môže na políčko vojsť, ak je voľné, je tam poklad, alebo tam sú mreže, ktoré vie otvoriť. Nemôže vojsť na políčko kde je stena alebo zamknuté mreže, od ktorých nemá kľúč.

Parížska kanalizácia sa nedá opustiť. Je navrhnutá tak, že keď Jacques vyjde z políčka na najľavejšom konci mapy, ocitne sa na políčku v tom istom riadku, ale v najpravejšom stĺpci a opačne. Podobne aj krok z prvého riadku nahor ho premiestni do posledného riadku toho istého stĺpca a naopak. Akonáhle Jacques nájde prvý poklad, tak je Charlottina úloha splnená.

Súťažná úloha

Napíšte program, ktorý načíta mapu kanalizácie (vrátane Jacquesovej začiatočnej pozície) a nájde najkratšiu cestu vedúcu k niektorému pokladu.

Je možné, že na mape žiaden poklad nie je, prípadne že sa Jacques k žiadnemu nebude vedieť dostať. Taktiež môže byť v kanáloch viac pokladov, viac mreží alebo kľúčov rovnakej farby.

Formát vstupu

Prvý riadok vstupu obsahuje dve prirodzené čísla R (počet riadkov) a S (počet stĺpcov) oddelené jednou medzerou. Môžete predpokladať, že $1 \leq R \cdot S \leq 1\,000\,000$. Každý z nasledujúcich riadkov obsahuje práve S znakov popisujúcich typy políčok na mape. Existujú tieto možné typy políčok:

`#`	stena
`.`	voľné políčko (chodba, miestnosť)
`*`	Jacquesova začiatočná pozícia – takéto políčko bude práve jedno
`CZMF`	zamknuté červené, zelené, modré alebo fialové mreže
`czmf`	červený, zelený, modrý alebo fialový kľúč
`$`	poklad

Mapa je vo vstupe orientovaná podľa zaužívaných konvencií, t.j. sever je hore.

Formát výstupu

Výstup má obsahovať jeden riadok a v ňom najkratšiu cestu, ktorou sa Jacques môže dostať k niektorému z pokladov. Ak je rovnako krátkych ciest viac, vypíšte ľubovoľnú z nich. Cestu vypisujte ako postupnosť znakov '`S`', '`J`', '`V`' a '`Z`', ktoré určujú, na ktorú svetovú stranu má Jacques spraviť krok. Ak neexistuje žiadna postupnosť spĺňajúca zadanie, tak namiesto popisu cesty vypíšte jeden riadok a v ňom jediné slovo '`nemozne`'.

Príklady

vstup

```
3 5
#####
#*F$#
#####
```

výstup

```
nemozne
```

Bez kľúča cez zamknutú fialovú mrežu neprejde.

vstup

```
3 19
###################
#$F.zMc.*.Cm.Z.ZZf#
###################
```

výstup

```
ZZVVVVVZZZZZZZVVVVVVVVVVVVVZZZZZZZZZZZZZZZZ
```

Jacques postupne vyzdvihne červený, modrý, zelený a fialový kľúč, v tomto poradí.

vstup

```
3 1
*
.
$
```

výstup

```
S
```

Prejde cez severný okraj.

Odovzdávanie riešení

Toto je praktická úloha. Odovzdávate **len funkčný, odladený program** prostredníctvom webového rozhrania.

A-I-3 Horská dráha

Na Vyšnej Klondike ešte minulý rok chodil vláčik. Kvôli problémom s rezervačným systémom však skrachoval a ostali len koľajnice.

Tie ležali len tak ladom a hrdzaveli, až kým jedného dňa neprišli opäť raz dvaja podnikaví zlatokopi, Santo a Banto, a nerozhodli sa, že na nich postavia pravú horskú dráhu.

Horská dráha musí začínať aj končiť v niektorej zo staníc železnice. Aby nebola jazda ňou ani príliš nudná, ani príliš nebezpečná, mal by mať jeden jej koniec práve o X metrov vyššiu nadmorskú výšku ako druhý. To sa však samozrejme nemusí dať dosiahnuť.

Súťažná úloha

Trať sa skladá z $N + 1$ staníc a N úsekov medzi nimi. Santo a Banto sa prešli pozdĺž nej a pre každý úsek i zistili hodnotu d_i, o ktorú sa na tomto úseku zmení nadmorská výška. (Ak je d_i kladné, trasa stúpa do kopca, ak je záporné, klesá, a ak je rovné nule, sú začiatok a koniec rovnako vysoko.)

Nájdite dve čísla i a j, pre ktoré je rozdiel nadmorských výšok konca j-teho a začiatku i-teho úseku čo najbližší k číslu X. Inými slovami, v postupnosti $d_1, \ldots, d_N$ nájdite tú súvislú podpostupnosť $d_i, \ldots, d_j$, ktorej súčet je S a hodnota $|X - S|$ je najmenšia možná. (Stačí nájsť jedno ľubovoľné optimálne riešenie.)

Príklad

vstup

```
N=5
X=7
3 5 -2 5 9
```

výstup

```
2 4
```

Indexom $i = 2$ a $j = 4$ zodpovedá podpostupnosť $5, -2, 5$, ktorej súčet je 8. Žiadna podpostupnosť nemá súčet 7, preto toto je jedna z optimálnych postupností.
Iné správne riešenia sú $i = 1$, $j = 2$ (súčet 8) a $i = 1$, $j = 3$ (súčet 6).

Odovzdávanie riešení

Toto je teoretická úloha. Riešenie, spĺňajúce požiadavky uvedené v pravidlách, buď pošlite poštou na adresu vašej krajskej komisie, alebo odovzdajte vo formáte PDF prostredníctvom webového rozhrania.

A-I-4 Zásobníkové počítače

V tomto ročníku olympiády sa budeme v teoretickej úlohe stretávať so zásobníkovými počítačmi. V študijnom texte uvedenom za zadaním tejto úlohy sú tieto stroje popísané.

Súťažná úloha

Reťazec voláme *palindróm*, ak sa číta rovnako odpredu aj odzadu – t. j., jeho prvý znak je rovnaký ako posledný, druhý ako predposledný, atď.

a) (5 bodov)
Napíšte program pre zásobníkový počítač, ktorý o zadanom reťazci rozhodne, či je to palindróm. Snažte sa dosiahnuť optimálnu časovú zložitosť.

b) (5 bodov)
Napíšte program pre zásobníkový počítač, ktorý o zadanom reťazci rozhodne, či je to palindróm. Snažte sa použiť minimálny počet zásobníkov.

Odovzdávanie riešení

Toto je teoretická úloha. Riešenie, spĺňajúce požiadavky uvedené v pravidlách, buď pošlite poštou na adresu vašej krajskej komisie, alebo odovzdajte vo formáte PDF prostredníctvom webového rozhrania.

Študijný text

V tomto ročníku olympiády si predstavíme zásobníkové počítače. Pamäť týchto počítačov je tvorená niekoľkými zásobníkmi. Každý zásobník obsahuje postupnosť hodnôt, s ktorou vie robiť tri operácie: Pridať novú hodnotu na koniec postupnosti (vložiť ju na vrch zásobníka), odstrániť hodnotu z konca postupnosti (vybrať ju zo zásobníka) a zistiť, či je postupnosť v zásobníku prázdna. Okrem zásobníkov vieme mať už len konštantný počet obyčajných premenných. Hodnoty uložené v zásobníkoch aj premenných musia mať vopred určený rozsah, ktorý nezávisí od veľkosti vstupu.

Naše zásobníkové počítače budeme programovať v jazyku Stackal. Ide o odrodu jazyka Pascal, upravenú podľa možností našich počítačov. V nasledujúcich odsekoch popíšeme, v čom sa náš jazyk od klasického Pascalu líši.

Premenné môžu byť týchto typov:
- `boolean` (logická premenná nadobúdajúca hodnoty `true` a `false`),
- `char` (premenná obsahujúca jeden znak),
- `a..b` (celočíselná premenná nadobúdajúca hodnoty od a po b vrátane, pričom $0 \leq a \leq b \leq 100$) a
- `stack of T`, kde `T` je typ iný ako `stack` (zásobník obsahujúci postupnosť hodnôt typu `T`).

Zásobníky sú na začiatku behu programu prázdne, obsah ostatných premenných je nedefinovaný.

Vstupom aj výstupom programu je postupnosť znakov – inými slovami, reťazec. Vstup čítame volaním funkcie `read(c)`, kde `c` je premenná typu `char`. Ak sme ešte vstup nedočítali, táto funkcia uloží do premennej `c` ďalšie písmeno zo vstupu a vráti `true`. V opačnom prípade vráti `false` a hodnotu premennej `c` nezmení. Na výstup zapíšeme znak z premennej `c` volaním procedúry `write(c)`. Na vstupe ani výstupe sa nie je možné vracať ani znaky preskakovať, program musí čítať vstup aj zapisovať výstup zaradom, „zľava doprava“.

Na prácu so zásobníkmi budeme mať špeciálne procedúry a funkcie. Nech `x` je premenná typu `T` a `S` je zásobník obsahujúci hodnoty tohto typu (teda `S` je premenná typu `stack of T`). Volaním procedúry `push(S,x)` vložíme do zásobníka `S` obsah premennej `x`. Funkcia `pop(S)` vyberie hodnotu zo zásobníka a vráti ju ako svoju návratovú hodnotu. Túto funkciu môžeme použiť aj ako procedúru, ak nás vybratá hodnota nezaujíma. Ak je pri volaní funkcie `pop(S)` zásobník `S` prázdny, program skončí s chybou – toto teda robiť nesmieme. Posledná užitočná funkcia je `empty(S)`, ktorá vráti `true` ak je zásobník `S` prázdny a `false` ak nie je. Žiadnym iným spôsobom nevieme s obsahom zásobníkov manipulovať.

K dispozícii máme všetky príkazy klasického Pascalu, môžeme si definovať aj vlastné pomocné procedúry a funkcie, avšak musíme dodržať niekoľko obmedzení. Je zakázané používať rekurziu. Priradzovací príkaz `:=` nesmieme použiť na zásobníky. Zásobník smieme dať ako parameter funkcii len odkazom, teda použitím kľúčového slova `var`.

(Najjednoduchšie riešenie, ako obmedzenia dodržať: Deklarujte všetky použité zásobniky na začiatku programu ako globálne premenné. Potom postupne definujte pomocné funkcie tak, aby každá z nich volala len funkcie definované skôr.)

Časovú a pamäťovú zložitosť definujeme podobne ako u klasických programov. Čas meriame počtom vykonaných príkazov, použitá pamäť je rovná maximálnemu počtu hodnôt, ktoré si program niekedy počas svojho behu pamätal v premenných a zásobníkoch.

Často bude našim cieľom minimalizovať počet použitých zásobníkov, a to aj za cenu horšej časovej zložitosti.

Príklad 1

Napíšte program, ktorý zistí, či je v zadanom reťazci rovnako veľa písmen `a` a `b`, a podľa toho vypíše na výstup buď znak `1` alebo znak `0`.

Riešenie 1a

Keďže rozsah celočíselných premenných je obmedzený, nemôžeme si počítať výskyty znakov `a` a `b` v premenných. Môžeme na to však ľahko využiť dva zásobníky. Do jedného si budeme ukladať `a`-čka a do druhého `b`-čka. Po dočítaní vstupu budeme súčasne vyberať znaky z oboch zásobníkov, a zistíme, či sa oba vyprázdnia naraz.

```
program rovnost;
var a, b: stack of char;                    { dva zásobníky na znaky }
    c: char;                                { práve spracúvaný znak }
begin
   while read(c) do begin                   { čítame zo vstupu, kým sa dá }
      if c='a' then push(a, c);             { znak vložíme do správneho zásobníku }
      if c='b' then push(b, c);
   end;
   while not empty(a) and not empty(b) do begin
      pop(a);                               { vyberáme znaky z oboch zásobníkov }
      pop(b);
   end;
   if empty(a) and empty(b) then write('1')  { sú prázdne oba? }
   else write('0');
end;
```

Tento algoritmus má lineárnu časovú aj pamäťovú zložitosť a potrebuje dva zásobníky.

Riešenie 1b

Na riešenie tejto úlohy stačilo použiť len jeden zásobník. Budeme si v ňom pamätať, o koľko viac znakov `a` ako znakov `b` sme doteraz prečítali. Presnejšie, ak bol tento rozdiel kladný, budeme mať v zásobníku príslušný počet znakov `+`, inak tam budeme mať príslušný počet znakov `-`.

Všimnime si napríklad, čo sa stane, keď zo vstupu prečítame ďalšie `a`. Rozdiel sa zvýšil o 1, potrebujeme teda upraviť zásobník. Skontrolujeme, aký znak je na jeho vrchu. Ak je to `-`, len ho odstránime. Ak je tam `+` alebo je zásobník prázdny, pridáme nové `+`. Znak `b` spracujeme opačne.

```
program rovnost2;

{ Pomocná funkcia, ktorá zistí znak na vrchu zásobníka }
function peek(var s:stack of char): char;
var c: char;
begin
   if empty(s) then c := '0'                { ak je prázdny, vrátime napr. nulu }
   else begin
      c := pop(s);                          { inak odoberieme prvok zo zásobníka }
```

```
      push(s, c);                          { vložíme ho hneď späť }
   end;
   peek := c;                              { a odovzdáme ho ako návratovú hodnotu }
end;

var r: stack of char;                      { tu je uložený rozdiel a-b }
    c: char;                               { práve spracúvaný znak }
begin
   while read(c) do begin
      if c='a' then                        { zvyšujeme rozdiel }
         if peek(r)='-' then pop(r) else push(r, '+');
      if c='b' then                        { znižujeme rozdiel }
         if peek(r)='+' then pop(r) else push(r, '-');
   end;
   if empty(r) then write('1')             { je rozdiel nulový? }
               else write('0');
end;
```

Na spracovanie každého znaku nám stačí konštantný počet príkazov, preto je časová zložitosť naďalej lineárna. V zásobníku bude najviac toľko znamienok, koľko je znakov na vstupe, preto je aj pamäťová zložitosť lineárna.

Adresy krajských komisií OI

Banskobystrický kraj
PaedDr. Miloslava Sudolská, KI FPV UMB, Tajovského 40, 974 01 B. Bystrica

Bratislavský kraj
RNDr. Eva Hanulová, Gymnázium Jura Hronca, Novohradská 3, 821 09 Bratislava

Košický kraj
RNDr. Rastislav Krivoš-Belluš, Ústav informatiky PF UPJŠ, Jesenná 5, 041 54 Košice

Nitriansky kraj
prof. Ing. Veronika Stoffová, CSc., KI PF UJS, Roľníckej školy 1519, 945 01 Komárno

Prešovský kraj
Mgr. Mária Majherová, Ph.D., Katedra matematiky, FHPV PU, Ul. 17 novembra č. 1, 081 16 Prešov

Trenčiansky kraj
Ing. Andrea Julény, Katedra informatiky, Fakulta mechatroniky TnUAD, Študentská 2, 911 50 Trenčín

Trnavský kraj
Mgr. Blanka Thomková, Gym. Jána Hollého, Na hlinách 30, 917 01 Trnava

Žilinský kraj
RNDr. Peter Varša, Ph.D., KI FRI ŽU, Moyzesova 20, 010 26 Žilina

SLOVENSKÁ KOMISIA OLYMPIÁDY V INFORMATIKE
DVADSIATY ŠTVRTÝ ROČNÍK OLYMPIÁDY V INFORMATIKE

Vydala IUVENTA s finančnou podporou Ministerstva školstva SR
Náklad: 400 výtlačkov
Zodpovedný redaktor: Michal Forišek
Sadzba programom LaTeX