MATEMATICKÁ OLYMPIÁDA NA STREDNÝCH ŠKOLÁCH

53. ročník, školský rok 2003/2004

Zadania úloh 1. kola kategórie P

Matematická olympiáda je súťaž pre študentov stredných škôl našej republiky. **Kategória P** je zameraná na programovanie a je určená študentom všetkých ročníkov.

Organizácia súťaže v kategórii P

Kategória P matematickej olympiády má tri postupové kolá — domáce, krajské a celoštátne.

V **I. kole** účastníci riešia štyri úlohy, uvedené v tomto letáku. Riešenia odovzdajú svojmu učiteľovi informatiky do **14. novembra 2003**. Učitelia informatiky odošlú riešenia v tomto termíne zo školy na príslušnú adresu podľa krajov takto:

Košický, Prešovský:
RNDr. G. Andrejková, CSc., KMI PF UPJŠ, Jesenná 5, 041 54 Košice
Žilinský:
RNDr. P. Varša, KI FRI Žilinská univerzita, Moyzesova 20, 010 26 Žilina
Banskobystrický:
Mgr. L. Huraj, KI FPV UMB, Tajovského 40, 974 01 Banská Bystrica
Trnavský, Trenčiansky, Nitriansky:
Doc. Ing. V. Štoffová, CSc., KI FPV UKF, tr. A. Hlinku 1, 949 74 Nitra
Bratislavský:
RNDr. A. Blaho, KVI FMFI UK, Mlynská dolina, 842 48 Bratislava 4

Najúspešnejší riešitelia domáceho kola sú pozvaní do **II. kola** (krajského), kde riešia štyri teoretické úlohy. Do **III. kola** (celoštátneho) sú pozývaní najúspešnejší riešitelia všetkých krajských kôl, pričom riešia tri teoretické úlohy a dve praktické úlohy pri počítači. Z najlepších riešiteľov tohto kola SK MO vyberie družstvá pre Medzinárodnú informatickú olympiádu a Stredoeurópsku informatickú olympiádu.

Predbežné termíny 53. ročníka MO, kategória P

I.	Domáce kolo	14. novembra 2003
II.	Krajské kolo	6. januára 2004
III.	Celoštátne kolo	apríl 2004

Usporiadateľ súťaže

Matematickú olympiádu vyhlasuje *Ministerstvo školstva SR* v spolupráci s *Jednotou slovenských matematikov a fyzikov* a *Slovenskou komisiou Matematickej olympiády*. Súťaž organizuje *Slovenská komisia MO* a v jednotlivých krajoch ju riadia *krajské komisie MO* pri pobočkách JSMF. Na jednotlivých školách ju zaisťujú učitelia matematiky a informatiky. Celoštátne kolo MO, tlač materiálov a ich distribúciu po organizačnej stránke zabezpečuje IUVENTA v tesnej súčinnosti so Slovenskou komisiou MO.

Formálna úprava riešenia

Riešenia súťažných úloh domáceho kola kategórie P pozostávajú z dvoch častí:

Popis riešenia. Riešenia musia obsahovať podrobný popis použitého algoritmu, **zdôvodnenie jeho správnosti** a diskusiu o efektivite zvoleného riešenia (t.j. posúdenie časových a pamäťových nárokov programu). Algoritmus by mal byť jasný už z popisu riešenia bez toho, aby bolo potrebné nahliadnuť do programu.

Program. V úlohách **P–I–1**, **P–I–2** a **P–I–3** je potrebné k riešeniu pripojiť odladený program napísaný v jazyku Pascal, C alebo C++. (Musí sa dať skompilovať kompilátorom *FreePascal*, resp. *gcc*.) Program sa odovzdáva v písomnej forme (jeho výpis je teda súčasťou riešenia) i na diskete, aby bolo možné otestovať jeho funkčnosť.

Súbory na diskete pomenujte `p1`x`.pas/.c/.cpp`, kde x je číslo súťažnej úlohy. Disketu označte menom riešiteľa. Z jednej školy možno poslať všetky riešenia na jednej diskete. V tomto prípade pre každého riešiteľa vytvorte podadresár označený jeho priezviskom a disketu označte adresou školy.

V úlohe **P–I–4** je potrebné uviesť program pre registrový počítač.

Písomnú časť riešenia vypracujte čitateľne na listy formátu A4. **Každú úlohu začnite na novom liste** a v záhlaví uveďte vaše meno, triedu, adresu školy a označenie príkladu podľa tohto letáku. Zadania úloh nemusíte opisovať. Ak sa vám riešenie nezmestí na jeden list, uveďte na ďalších listoch vľavo hore svoje meno a označenie úlohy a očíslujte strany.

P-I-1

Firma Pripojsa poskytovala svojim zákazníkom veľmi spoľahlivé pripojenie na internet. Kvôli tomu si vybudovala svoju vlastnú sieť spájajúcu niekoľko najväčších miest krajiny. Sieť pozostáva z uzlov umiestnených v týchto mestách a z liniek vedúcich medzi mestami. Každá linka spája dve mestá. Táto sieť mala tú vlastnosť, že ak sa prerušila ľubovoľná linka, sieť stále zostala funkčná, t.j. zvyšné linky umožňovali komunikáciu medzi ľubovoľnými dvoma mestami.

Nedávno sa firma rozhodla rozšíriť svoje služby aj pre zákazníkov v ďalších mestách. Preto vybudovala množstvo liniek pripájajúcich tieto mestá k už existujúcej sieti. Teraz by vo firme potrebovali vedieť, či je ich sieť stále ešte dostatočne spoľahlivá.

Súťažná úloha

Napíšte program, ktorý načíta zoznam uzlov a liniek siete a zistí, či má sieť tú vlastnosť, že ak sa ľubovoľná jedna linka preruší, všetky mestá v sieti môžu ešte stále medzi sebou komunikovať pomocou zvyšných liniek.

Formát vstupu. Prvý riadok vstupného súboru `siet.in` obsahuje dve kladné celé čísla n a m, kde n je počet miest v sieti a m je počet liniek ($n \leq 100$). Pre jednoduchosť sú mestá v sieti očíslované číslami $1, 2, \ldots, n$. Na každom z nasledujúcich m riadkov sú dve čísla určujúce mestá spojené linkou.

Môžete predpokladať, že je možné komunikovať medzi každými dvomi mestami v sieti a že žiadna dve linky nespájajú rovnakú dvojicu miest.

Formát výstupu. Výstupný súbor `siet.out` obsahuje jediný riadok, na ktorom je slovo "ANO", ak sa v sieti na vstupe dá komunikovať medzi ľubovoľnými dvoma mestami aj po prerušení ľubovoľnej (jednej) linky a slovo "NIE", ak sieť túto vlastnosť nemá.

Príklad:

Súbor `siet.in`

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

Súbor `siet.out`

```
ANO
```

Súbor `siet.in`

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

Súbor `siet.out`

```
NIE
```

(Ak sa preruší linka medzi mestami 3 a 4, tieto mestá nebudú môcť komunikovať.)

P-I-2

Janko si založil maličkú programátorskú firmu, ktorú nazval príznačne AttoSoft.[1] Nedávno sa mu konečne podarilo získať prvého klienta. Ten mu dal za úlohu naprogramovať N jednoduchých programov.

Janko je však lenivý programovať, preto si najal N programátorov. Keď ráno všetci prišli do práce, ukázalo sa, že každý z nich vie naprogramovať iba jeden z potrebných programov. Našťastie každý z potrebných programov vedel niekto z nich naprogramovať, takže neostávalo nič iné, len sa pustiť do práce. A tu vznikol problém. AttoSoft totiž vlastní len jeden počítač a na ňom v jednom okamihu môže robiť len jeden programátor.

Zrazu si Janko uvedomil, že nie je jedno, v akom poradí budú programátori písať svoje programy. Podľa podpísanej zmluvy totiž programátorov neplatí za vykonanú prácu. Každého programátora platí za čas, ktorý uplynie od začiatku práce na celej zákazke až do okamihu, kým ten programátor nedokončí svoj program. Janko o každom z programátorov vie, koľko mu musí zaplatiť za jednu hodinu a ako dlho mu napísanie jeho programu bude trvať. Pomôžte mu zistiť, v akom poradí má programátorov poslať programovať, aby im dokopy zaplatil čo najmenej.

Príklad. Majme troch programátorov A, B, C. A chce 100 Sk za hodinu a na svoj program potrebuje 2 hodiny. B chce 20 Sk za hodinu a potrebuje 5 hodín času. C chce 500 Sk za hodinu a potrebuje 20 hodín.

Ak by programovali v poradí A, B, C, tak by Janko musel zaplatiť A za 2 hodiny, B za 7 a C za 27 hodín, čo by ho vyšlo na 13 840 Sk.

Ak by programovali v poradí C, B, A, stálo by to Janka len $500 \times 20 + 20 \times 25 + 100 \times 27 = 13\,200$ Sk, preto takéto poradie je lepšie. (Ale nie je to najlepšie možné riešenie.)

Súťažná úloha

Napíšte program, ktorý načíta počet programátorov, ich hodinové mzdy a časy potrebné na ich prácu a spočíta, v akom poradí má Janko nechať programátorov pracovať, aby im dokopy zaplatil najmenšiu možnú sumu.

[1] Mikro je 10^{-6}, piko je 10^{-12}, atto je 10^{-18}.

Formát vstupu. Prvý riadok vstupného súboru `attosoft.in` obsahuje kladné celé číslo N ($1 \leq N \leq 10\,000$), kde N je počet programátorov. Nasleduje N riadkov, i-ty z nich obsahuje dve celé čísla m_i, t_i ($0 < m_i, t_i \leq 30\,000$), kde m_i je hodinová mzda i-teho programátora a t_i je čas v hodinách, ktorý i-ty programátor potrebuje na napísanie svojho programu.

Formát výstupu. Výstupný súbor `attosoft.out` obsahuje N riadkov, v j-tom z nich je jedno celé číslo od 1 do N – číslo programátora, ktorý má programovať ako j-ty. Ak má úloha viac riešení, vypíšte ľubovoľné z nich.

Príklad:

Súbor `attosoft.in`	**Súbor** `attosoft.out`
`3`	`1`
`100 2`	`3`
`20 5`	`2`
`500 20`	*(Janko zaplatí* 11 740 *Sk.)*

P-I-3

Dané je pole $A[1..N]$ celých čísel. Napíšte program, ktorý bude vedieť čo najrýchlejšie spracúvať nasledujúce príkazy:
– zmeň hodnotu $A[x]$ na y
– vypíš súčet prvkov $A[x] + A[x+1] + \ldots + A[y]$
Váš program si na začiatku môže pole A v rozumnom čase predspracovať.

Formát vstupu. Vstupný textový súbor `sucty.in` obsahuje vopred neurčený počet riadkov. Na prvom riadku je uvedené jediné celé číslo N ($1 \leq N \leq 2\,000$). Druhý riadok obsahuje pôvodné hodnoty v poli A – celé čísla $a_1, a_2, \ldots, a_N$ oddelené medzerami ($0 \leq a_i \leq 1\,000\,000$).

Nasleduje niekoľko riadkov s príkazmi, každý z nich je jedného z nasledujúcich tvarov:
$1\ x\ y$ ($1 \leq x \leq N$, $0 \leq y \leq 1\,000\,000$) zmeň hodnotu $A[x]$ na y
$2\ x\ y$ ($1 \leq x \leq y \leq N$) vypíš hodnotu $A[x] + \ldots + A[y]$
Vstup je ukončený riadkom obsahujúcim jediné číslo 0.

Formát výstupu. Váš program musí spracovať príkazy v poradí, v akom sú uvedené na vstupe. Pre každý príkaz na vypísanie súčtu nejakých prvkov poľa musí do výstupného súboru `sucty.out` zapísať jeden riadok a na ňom jedno celé číslo – súčet príslušných prvkov poľa v danom okamihu.

Príklad:

Vstup	**Výstup**
14	26
1 4 3 1 1 3 1 1 3 1 4 1 1 1	8
2 1 14	8
2 3 6	5
1 2 0	
2 3 6	
1 3 0	
2 3 6	
0	

P-I-4

V tejto úlohe sa budeme zaoberať *registrovými počítačmi*. *Register* je niečo podobné ako premenná. Presnejšie, v registri môže byť uložené *ľubovoľne veľké* nezáporné celé číslo. Na rozdiel od premenných, ktoré vieme medzi sebou sčítať, odčítať a násobiť, s registrom vieme robiť len tri jednoduché operácie: zväčšiť jeho obsah o 1, zmenšiť jeho obsah o 1 (ak v ňom bola 0, ostane v ňom 0) a pozrieť sa, či je v ňom 0.

Registrový počítač má k dispozícii neobmedzene veľa registrov, označených R_0, R_1, R_2, atď. Okrem registrov má registrový počítač k dispozícii len **konečne veľkú** pomocnú pamäť.

Program pre náš registrový počítač budeme písať v jazyku podobnom jazyku Pascal. Náš programovací jazyk bude rozšírený napr. o príkazy na prácu s registrami, na druhej strane niektoré príkazy z Pascalu budeme musieť zakázať.

Náš počítač bude riešiť úlohy nasledovného typu: dostane na vstupe slovo (reťazec písmen) a po nejakom čase odpovie, či je to slovo dobré alebo nie. Aby nám vedel odpovedať, zavedieme špeciálne príkazy **Accept** a **Reject**. Akonáhle sa vykoná príkaz **Accept**, slovo je dobré a výpočet končí. Akonáhle sa vykoná príkaz **Reject**, slovo je zlé a výpočet končí. Ak sa výpočet zacyklí, prípadne ak skončí a počas neho sa nevykoná ani **Accept**, ani **Reject**, slovo je tiež zlé.

Príkaz "prirátaj 1 k obsahu registra R" budeme značiť **Inc**(R), "odrátaj 1 od obsahu registra R" budeme značiť **Dec**(R). Výraz **Zero**(R) bude pravdivý, ak je v registri R nula a nepravdivý inak. Na začiatku výpočtu sú vo všetkých registroch nuly.

V programe môžeme použiť len konečne veľa registrov. Okrem nich môžeme použiť už iba konštantný počet pomocných premenných typu *byte*[2]

[2]obsah takejto premennej je celé číslo od 0 do 255

(teda nemôžeme použiť polia!) a špeciálnu premennú **vstup** typu *char*. Premennú **vstup** môžeme meniť len zavolaním príkazu **Read**(**vstup**). Ak počítač ešte nedočítal vstupné slovo, príkaz **Read**(**vstup**) prečíta ďalšie písmeno a uloží ho do premennej **vstup**. Ak počítač už vstupné slovo dočítal, príkaz **Read**(**vstup**) uloží do premennej **vstup** špeciálny znak \$.

Keďže registrový počítač má okrem registrov len konečne veľa pamäte, nemôže si dovoliť používať ani rekurziu (nemal by si kde pamätať návratové adresy). My pre istotu úplne zakážeme definovať a používať v našom programe procedúry. Takisto je zakázané volanie štandardných procedúr a funkcií jazyka Pascal. V aritmetických výrazoch sa smú používať iba premenné (teda nie registre), konštanty, celočíselné operátory $+$, $-$, $*$, div, mod a zátvorky. V podmienkach sa smú používať výrazy **Zero**(R_i), bežné relačné operátory ($<$, $<=$, ...), logické spojky a zátvorky.

Z kľúčových slov jazyka Pascal sú teda povolené iba nasledovné: **var**, **begin**, **end**, **if**, **then**, **else**, **case**, **of**, **while**, **do**, **repeat**, **until**, **for**, **to**, **downto**, **div**, **mod**, **and**, **or**, **not** a **xor**.

Príklad 1

Napíšte program pre registrový počítač, ktorý bude riešiť nasledujúcu úlohu: Na vstupe bude reťazec písmen a, b, c. Nech α je počet tých písmen a vo vstupnom reťazci, za ktorými už nie je žiadne c. Podobne nech β je počet b, za ktorými nie je žiadne c. Počítač má vstupný reťazec vyhlásiť za dobrý práve vtedy, keď $\alpha = \beta$.

Riešenie. V registri R_1 si pamätáme aktuálnu hodnotu α, v registri R_2 hodnotu β. Vždy, keď prečítame písmeno c, oba registre vynulujeme. Na konci jednoducho porovnáme hodnoty uložené v registroch. Rozmyslite si, že by stačilo použiť jeden register (v ktorom by sme mali hodnotu $|\alpha - \beta|$).

```
var vstup : char;
begin
  Read(vstup);
  while (vstup <> '$') do begin
    if (vstup = 'a') then Inc(R1);
    if (vstup = 'b') then Inc(R2);
    if (vstup = 'c') then begin
      while not Zero(R1) do Dec(R1);
      while not Zero(R2) do Dec(R2);
    end;
  end;
  while not Zero(R1) do begin
    Dec(R1);
    if (Zero(R2)) then Reject;
```

```
    Dec(R2);
  end;
  if Zero(R2) then Accept;
end.
```

Príklad 2

Napíšte program pre registrový počítač, ktorý bude riešiť nasledujúcu úlohu: Na vstupe bude reťazec písmen a. Počítač ho má vyhlásiť za dobrý práve vtedy, keď je jeho dĺžka mocninou troch.

Riešenie. Prečítame vstupné slovo, pričom si do R_1 uložíme jeho dĺžku. Potrebujeme zistiť, či je to mocnina troch. Aby sme to zistili, budeme uloženú hodnotu deliť tromi, kým to pôjde. Ak na konci dostaneme podiel 0 a zvyšok 1, pôvodné číslo bola mocnina 3, inak nie.

```
var vstup : char;
    zvysok : byte;
begin
  Read(vstup);
  while (vstup <> '$') do begin Inc(R1); Read(vstup); end;
  if Zero(R1) then Reject;
  while true do begin
    zvysok := 0;
    while not Zero(R1) do begin
      Dec(R1);
      zvysok := (zvysok + 1) mod 3;
      if (zvysok = 0) then Inc(R2);
    end;
    if (Zero(R2) and (zvysok = 1)) then Accept;
    if (zvysok <> 0) then Reject;
    while not Zero(R2) do begin Dec(R2); Inc(R1); end;
  end;
end.
```

Súťažná úloha

Napíšte program pre registrový počítač, ktorý bude riešiť nasledujúcu úlohu: Vstupom bude reťazec písmen a, b, c, d. Počítač ho má vyhlásiť za dobrý práve vtedy, keď je v ňom najviac písmen a.

Teda napríklad vstup *baacd* je dobrý, vstup *baacdbb* nie je dobrý (lebo písmen b je viac ako a) a ani vstup *ac* nie je dobrý (lebo písmen a a c je rovnako).

Základným kritériom hodnotenia bude počet registrov, ktoré váš počítač potrebuje. Snažte sa zostrojiť počítač, ktorý ich používa čo najmenej!

SLOVENSKÁ KOMISIA MATEMATICKEJ OLYMPIÁDY

53. ROČNÍK MATEMATICKEJ OLYMPIÁDY

Zadania 1. kola kategórie P

Vydala IUVENTA
pre vnútornú potrebu Ministerstva školstva SR
Náklad: 300 výtlačkov
Autori úloh: B. Brejová, M. Forišek, M. Pál
Zodpovedný redaktor: M. Forišek
Sadzba programom LaTeX