MATEMATICKÁ OLYMPIÁDA NA STREDNÝCH ŠKOLÁCH

53. ročník, školský rok 2003/2004

Zadania úloh 3. kola kategórie P

1. súťažný deň

Na riešenie úloh máte 4.5 hodiny čistého času. Riešenie každého príkladu musí obsahovať (pokiaľ nie je v zadaní uvedené ináč):

- **Popis riešenia,** to znamená slovný popis použitého algoritmu, argumenty zdôvodňujúce jeho správnosť (prípadne dôkaz správnosti algoritmu), diskusiu o efektivite vášho riešenia (časová a pamäťová zložitosť). Slovný popis riešenia musí byť jasný a zrozumiteľný i bez nahliadnutia do samotného zápisu algoritmu (do programu).
- **Program.** V úlohách **P-III-1** a **P-III-2** treba uviesť dostatočne podrobný zápis algoritmu, najlepšie v tvare zdrojového textu najdôležitejších častí programu v jazyku Pascal alebo C. Zo zápisu môžete vynechať jednoduché operácie ako vstupy, výstupy, implementáciu jednoduchých matematických vzťahov a pod. V úlohe **P-III-3** uveďte príslušné programy pre registrové počítače.

Hodnotí se nielen správnosť programu, ale tiež kvalita popisu riešenia a efektivita zvoleného algoritmu.

P-III-1

Istá nemenovaná tajná spoločnosť má N agentov. V záujme utajenia každý agent môže vydávať rozkazy iba niekoľkým ďalším agentom. Agent, ktorý dostane rozkaz, pošle tento rozkaz všetkým agentom, ktorým môže vydávať rozkazy. Šéfom spoločnosti je taký agent, ktorý ak vydá rozkaz, tak ho časom dostanú všetci agenti. (Spoločnosť môže mať aj viacero šéfov, nemusí mať žiadneho.)

Súťažná úloha

Na vstupe je počet agentov N. Agenti sú očíslovaní číslami od 7 (presnejšie 007) do $N + 6$. Pre každého agenta je tiež daný zoznam agentov, ktorým môže vydať rozkaz. Navrhnite efektívny algoritmus, ktorý nájde šéfa tajnej spoločnosti (ak je ich viac, stačí nájsť ľubovoľného) alebo zistí, že tajná spoločnosť nemá šéfa.

Príklad:

Vstup:
$N = 3$
agent 7 rozkazuje agentovi 8
agent 8 rozkazuje agentovi 9
agent 9 rozkazuje agentovi 7

Výstup:
Agent 7 je šéf.

Vstup:
$N = 4$
agent 7 nerozkazuje nikomu
agent 8 nerozkazuje nikomu
agent 9 rozkazuje agentom 7 a 8
agent 10 rozkazuje agentom 7 a 8

Výstup:
Žiaden agent nie je šéf.

P-III-2

Meteorologická stanica každú minútu meria teplotu vzduchu. Meteorológovia by potrebovali program, ktorý by im v každom okamihu hovoril, aká najnižšia teplota bola nameraná počas posledných K minút. Vašou úlohou bude napísať tento program.

Na vstupe je číslo K, nasleduje postupnosť nameraných teplôt ukončená hodnotou -1000. Váš program by vždy po prečítaní teploty mal vypísať najnižšiu spomedzi posledných K načítaných teplôt.[1]

Príklad:

Vstup	**Výstup**
$K = 3$	
teploty:	
9.0	9.0
4.7	4.7
5.3	4.7
2.1	2.1
9.0	2.1
9.8	2.1
17.0	9.0
9.5	9.5
-1000	

[1] Resp. najnižšiu spomedzi doteraz načítaných teplôt, ak ich bolo menej ako K.

P-III-3

V tomto kole sa budeme zaoberať tzv. *jednosmernými registrovými počítačmi* – rovnakými ako v domácom kole. Ich formálnu definíciu nájdete v študijnom texte, ktorý nasleduje za zadaním súťažnej úlohy.

Súťažná úloha

a) Nech R je reťazec tvorený písmenami a, b, c, A, B, C. Označme $m(R)$ reťazec tvorený malými písmenami v R (v poradí, v akom sa vyskytujú v R). Analogicky označme $v(R)$ reťazec tvorený veľkými písmenami v R. Reťazec $upcase(R)$ dostaneme z R tak, že nahradíme všetky malé písmená zodpovedajúcimi veľkými.

   Napr. ak $R = aaAcB$, tak $m(R) = aac$, $v(R) = AB$ a $upcase(R) = AAACB$.

   Napíšte program pre jednosmerný registrový počítač, ktorý bude riešiť nasledujúcu úlohu: Na vstupe dostane reťazec R, tvorený písmenami a, b, c, A, B, C. Počítač ho má vyhlásiť za dobrý práve vtedy, ak $upcase(m(R)) = v(R)$. (Slovom: ak veľké písmená v R tvoria "to isté" slovo ako malé.)

   Teda napr. vstupy aA, Aa a $abAcBaCABb$ sú dobré, ale vstupy aa, $BcbC$ ani $acACa$ nie sú dobré.

   Váš program môže použiť ľubovoľný konečný počet registrov, hodnotí sa len jeho správnosť. Ak si myslíte, že takýto program neexistuje, dokážte to.

b) Dokážte, že pre ľubovoľnú úlohu platí: Ak vieme zostrojiť program registrový počítač, ktorý rieši danú úlohu pomocou troch registrov, tak vieme zostrojiť program, ktorý túto úlohu rieši pomocou iba dvoch registrov.

   Inými slovami: Ukážte postup, ako existujúci program používajúci 3 registre prepísať na ekvivalentný, ktorý potrebuje len dva registre. Nezabudnite zdôvodniť správnosť vášho postupu.

Študijný text

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] (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**, **ord**, **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

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

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.
```

SLOVENSKÁ KOMISIA MATEMATICKEJ OLYMPIÁDY

53. ROČNÍK MATEMATICKEJ OLYMPIÁDY

Zadania 3. kola kategórie P

1. súťažný deň

Vydala IUVENTA
pre vnútornú potrebu Ministerstva školstva SR
Náklad: 50 výtlačkov
Autori úloh: B. Brejová, M. Forišek, M. Pál
Zodpovedný redaktor: M. Forišek
Sadzba programom LaTeX

MATEMATICKÁ OLYMPIÁDA NA STREDNÝCH ŠKOLÁCH

53. ročník, školský rok 2003/2004

Zadania úloh 3. kola kategórie P

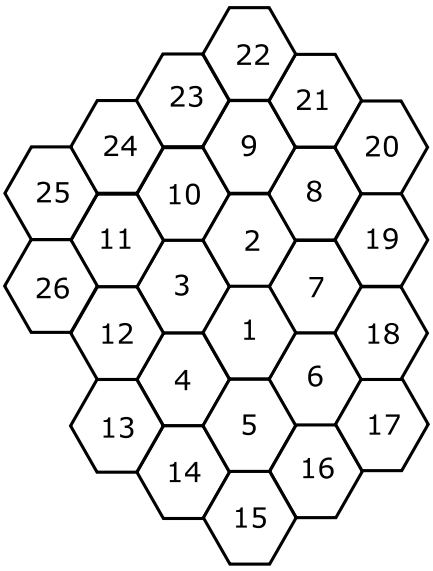
2. súťažný deň

P-III-4

Program: `psici.pas/.c/.cpp`
Vstup: `psici.in`
Výstup: `psici.out`

"Ták, a teraz budete skátať vždy, keď zapískam! A každý ináč." rozkázal Konrád svojim dvom psíkom. Chudáci malí, musia teraz obidvaja poskakovať po lúke, až kým sa obidvaja súčasne neschovajú.

Svet je nekonečná šesťuholníková sieť. Políčka sveta sú očíslované prirodzenými číslami idúc od čísla 1 po špirále. Lúku tvorí prvých N políčok sveta. Na obrázku je príklad lúky pre $N = 26$.



Na lúke stoja naši dvaja psíci na políčkach S_1, S_2. Skrýš pre prvého je na políčku T_1, pre druhého na políčku T_2. Na M políčkach lúky rastú bodliaky a psíci tam za žiadnych okolností neskočia.

Na jedno písknutie dokážu psíci preskočiť na ľubovolné susedné políčko, pokiaľ na ňom nerastú bodliaky. Nemôžu skákať obidvaja súčasne tým istým smerom a taktiež nemôžu obidvaja dopadnúť na to isté políčko.

Na každé písknutie musí každý z nich preskočiť na iné políčko (aj keby už stál v skrýši).

Vašou úlohou je zistiť, na koľko najmenej písknutí môžu psíci stáť súčasne vo svojich skrýšach.

Súťažná úloha

Na vstupe je za sebou popis niekoľkých (max. 5) problémov.

Každý problém má na prvom riadku 2 čísla N ($2 \leq N \leq 500$), M ($0 \leq M \leq N-2$), na druhom riadku čísla políčok S_1, T_1, S_2 a T_2 ($1 \leq S_1, T_1, S_2, T_2 \leq N$). Nasleduje ďalších M riadkov s číslami políčok, kde rastú bodliaky.

Vstup je ukončený riadkom obsahujúcim $M = N = 0$.

Môžete predpokladať, že $S_1 \neq S_2$, $T_1 \neq T_2$ a že na políčkach S_1, S_2 nie sú bodliaky. Za posledným vstupom je riadok s dvoma nulami.

Na výstup vypíšte pre každý problém jeden riadok s najmenším počtom písknutí, po ktorom môžu psíci súčasne stáť vo svojich skrýšach. Ak sa to nedá, vypíšte namiesto počtu taktov text "neda sa".

Príklad:

Vstup:	**Výstup:**

`11 0`
`3 10 6 7`
`11 1`
`3 10 6 7`
`1`
`10 2`
`3 10 7 8`
`2`
`9`
`0 0`
	`2`
	`3`
	`neda sa`

P-III-5

Jankovej programátorskej firme AttoSoft sa podarilo získať ďalšieho klienta, ktorý potrebuje naprogramovať N programov. Jankovi a jeho programátorom sa však do práce veľmi nechcelo a tak prišiel termín odovzdania a programy ešte stále nie sú hotové. Janko sa zľakol a začal študovať zmluvu, ktorú so zákazníkom podpísal.

V zmluve bol pre každý program uvedený vzorec, akým sa počíta pokuta za neskoré odovzdanie programu v závislosti od doby omeškania. Našťastie netreba zaplatiť súčet pokút, **ale len najvyššiu pokutu zo**

všetkých. Teraz sa Janko snaží naplánovať prácu na programoch tak, aby zaplatil čo najmenšiu pokutu. Ako predtým, aj teraz má k dispozícii len jeden počítač, a preto sa nedá pracovať na viacerých programoch naraz. Prácu na jednom programe nemožno prerušiť.[3]

Súťažná úloha

Na vstupe je pre každý z nedokončených programov vzorec na výpočet pokuty a koľko dní treba na jeho dokončenie. Napíšte program, ktorý nájde rozvrh na dokončenie programov, pri ktorom Janko zaplatí najmenšiu pokutu. Pre jednoduchosť má vzorec na výpočet pokuty tvar polynómu najviac tretieho stupňa ax^3+bx^2+cx+d, v ktorom sú koeficienty a, b, c, d nezáporné a x je počet dní, o ktoré sa odovzdanie programu omeškalo.

Formát vstupu. Prvý riadok vstupného súboru obsahuje kladné celé číslo N ($1 \leq N \leq 5\,000$) – počet programov. Nasleduje N riadkov, i-ty z nich obsahuje päť celých čísel l_i, a_i, b_i, c_i, d_i ($1 \leq l_i \leq 100$, $0 \leq a_i, b_i, c_i, d_i \leq 5\,000$) kde l_i je čas v dňoch potrebný na dokončenie i-teho programu a a_i, b_i, c_i, d_i sú koeficienty vzorca na výpočet pokuty. Môžete predpokladať, že za 100 000 dní sa stihnú naprogramovať všetky programy.

Formát výstupu. Výstupný súbor obsahuje N čísel, oddelených bielymi znakmi (medzerami alebo koncami riadkov). Tieto čísla reprezentujú čísla programov v poradí, v akom ich treba dokončiť, aby pokuta bola najmenšia možná. Ak má úloha viac riešení, vypíšte ľubovoľné z nich.

Príklad:

Vstup	**Výstup**
3 10 1 0 0 0 3 0 0 0 10 1 0 0 5 0	1 3 2

Poznámka. Pokuta za program číslo 1 dokončený po desiatich dňoch je $10^3 = 1000$, za program číslo 2 dokončený po 14 dňoch je 10 a za program číslo 3 dokončený po 11 dňoch je $5 \cdot 11 = 55$. Janko teda zaplatí pokutu 1000.

[3] Šikovnejší z vás si po prečítaní zvyšku zadania uvedomia, že aj keby sa to smelo, neoplatí sa to.

SLOVENSKÁ KOMISIA MATEMATICKEJ OLYMPIÁDY

53. ROČNÍK MATEMATICKEJ OLYMPIÁDY

Zadania 3. kola kategórie P

2. súťažný deň

Vydala IUVENTA
pre vnútornú potrebu Ministerstva školstva SR
Náklad: 50 výtlačkov
Autori úloh: B. Brejová, M. Forišek, M. Pál
Zodpovedný redaktor: M. Forišek
Sadzba programom LaTeX