Korešpondenčný seminár z programovania
XXI. ročník, 2003/04
Katedra vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

KSP finančne podporuje MICROSTEP-MIS spol. s r.o.

Príklady 1. kola zimnej časti

Milé naše riešiteľky, mílí naši riešitelia,

Srdečne vás všetkých pozývame riešiť ďalší ročník Korešpondenčného seminára z programovania. Tešíme sa na kopu skvelých riešení, ktorými nás aj tento rok zavalíte. A vy sa zase môžete tešiť na úžasné sústredenia, ktoré každého pol roka organizujeme pre najlepších riešiteľov. Ale na to, aby ste sa na sústredenie dostali (a samozrejme na to, aby sme vás čo-to z programovania naučili), musíte najskôr vyriešiť úlohy, ktoré nájdete v tomto letáku. Svoje riešenia nám posielajte najneskôr do **20. októbra 2003**. Nezabudnite k nim priložiť známky v hodnote 15,- Sk.

Ne auderis delere orbem rigidum meum![1]

KSPáci

1. O slovných rovniciach

15 bodov

Keďže Kleofáš[2] je šikovný žiačik a navyše minulý rok sa v škole naučil písať, jeho mamička si myslela, že mu urobí radosť, keď mu na narodeniny namiesto starého bicykla od susedov kúpi novú zbierku takzvaných slovných rovníc. Hm, tak nie... :-)

Ľudia robia chyby, tak to už chodí. Výsledkom sú potom smutní Kleofášovia ako sedia za stolom, pred sebou akási zbierka, hrubý zošit otvorený na prvej strane a pred sebou okno s výhľadom na bicyklujúcich sa kamarátov. Ešteže sú na svete starší bratia, z ktorých niektorí riešia KSP...

Úloha: Slovná rovnica sa podobá obyčajnej lineárnej rovnici s jednou premennou. Má tvar $u = v$, kde u a v sú slová zložené z písmen abecedy a slovnej premennej X, pričom počet výskytov X je na každej strane rovnice iný.

Riešenie slovnej rovnice (ak existuje) je také slovo w, že keď ním nahradíme všetky výskyty premennej X na ľavej aj pravej strane, prečítame na oboch stranách rovnaké slovo.

Vašou úlohou je napísať program, ktorý na vstupe dostane slovnú rovnicu a vypíše jej jedno riešenie (prípadne povie, že daná rovnica nemá riešenie).

Príklad:

Vstup:	**Výstup:**
$abacXba = XcabX$	$X = aba$
$abXd = aXcX$	Rovnica nemá riešenie.

2. O trpaslíčom probléme

15 bodov

U trpaslíkov v dedine opäť nie je všetko úplne v poriadku. Ako je široko ďaleko známe, v každom domčeku v dedine bývajú nejakí trpaslíci. Domčeky sú pospájané cestičkami, to aby sa mohli trpaslíci navštevovať. Po cestičkách sa dá (aj keď nie nutne priamo) dostať z ľubovoľného domčeka do ľubovoľného iného. Keď sa však trpaslík vracia z návštevy, musí

[1] latinsky: Neopováž sa zmazať môj harddisk!

[2] Všetky mená, príbuzenské vzťahy a pointa príbehu boli zmenené. Poučenie a zadanie však ostáva nezmenené.

ísť naspäť po tej istej ceste – medzi každými dvoma domčekmi sa dá prejsť len jedným spôsobom.[3]

Trpaslík Hubert je podnikavý typ trpaslíka, preto si už dávnejšie prerobil časť domčeka na obchod. Viac obchodov by sa v dedine neuživilo, a tak chodia všetci trpaslíci nakupovať výlučne k Hubertovi. Problémy však vznikli potom, ako sa do módy dostali motorové trojkolky. Každý trpaslík si jednu zaobstaral (pre Huberta to boli zlaté časy) a aby len tak neležala v garáži, každý deň sa na nej vybral do obchodu. Mamky trpaslice robili rodinné nákupy, oteckovia trpaslíci si ráno zašli po noviny, trpaslíčatá ozlomkrky uháňali po sladkosti, len čo si dopísali úlohy...

Používať motorovú trojkolku okrem iného znamená aj znečisťovať ovzdušie. Čím ďalej zájdete, tým viac znečistíte ovzdušie. Presnejšie, n-členná rodina bývajúca d metrov od obchodu vypustí do ovzdušia $47 \cdot n \cdot d$ centimetrov kubických škodlivých plynov (údaje sú uvedené na obale od trojkolky). Starosta dediny preto z obáv o čistý vzduch rozhodol, že obchod už asi nebude u Huberta doma, ale bude v takom domčeku, že denné znečistenie ovzdušia obyvateľmi dediny bude najmenšie možné. Úloha sa však zdá byť nad starostove sily, preto mu radšej pomôžte.

Úloha: Na vstupe je popísaná trpasličia dedina. Na prvom riadku je celé číslo N, ktoré určuje počet domčekov (označených číslami od 1 po N) v dedine. Potom nasleduje N čísel $a_1 \ldots a_N$, kde a_i je počet trpaslíkov v i-tom domčeku. Ďalej nasleduje $N-1$ riadkov, v každom z nich tri čísla i, j, d, znamenajúce, že medzi domčekmi i a j vedie priama cesta dĺžky d. (Rozmyslite si, že cestičiek je v dedine práve $N-1$.) Váš program by mal vypísať číslo domčeka, v ktorom by mal byť obchod.

Príklad:

Vstup:
$N = 5$
2 7 4 3 5
1 3 10
1 4 5
1 5 7
2 4 1

Výstup:
Obchod je treba dať do domčeka 1.

3. Ohne druidov

15 bodov

Jožko znova vyletel z analýzy. Kráčal zahmlenou Bratislavou a bol nekonečne nešťastný. Zrazu sa zem zdvihla a udrela ho do čela. Jožko si vo svojom žiali nevšimol jahodu, čo rástla pri Matfyze. Keď sa prebral, okolo neho neboli budovy milovanej školy, ale stromy, tráva a nekonečná divočina. Na sebe mal rytierske brnenie a v ruke meč. Neveriacky naň hľadel. Po čase prišiel na to, že sa nachádza na území kráľa Artuša. Potreboval sa však dostať späť na opravný termín z analýzy. Bezradne sa pohyboval lesom a premýšľal. Zdvihol pohľad, uvidel chalupu a pri nej starca. Mal dlhú bradu a na nechtoch pleseň. „Keby to tak bol druid...“ pomyslel si Jožko. Druidi boli mocní zaklínači, vedeli komunikovať s duchmi sveta a keby zapálili svoj čarovný oheň a dali sily dokopy, vedeli by napríklad aj dostať Jožka späť do Bratislavy. Pomaly sa mu začala vracať nádej. Zapísal si polohu domu. Obehol znova celý les dookola a našiel ešte ďalších pár chalúp a starcov. Teraz ostáva zistiť, či sa nikde nezmýlil.

Druidi počas zaklínania stoja vo vrcholoch pravidelného $2N$-uholníka, v ktorého strede horí oheň. Keď skončia zaklínanie, rozídu sa do svojich domov. Pritom sa ale každý druid pohybuje len po polpriamke, ktorá vedie z ohniska cez miesto, kde stál počas zaklínania. Jožko by potreboval zistiť, či domce, ktoré našiel, môžu byť práve všetky domy druidov. Lenže analýzu nespravil, a tak analyzovať nevie...

[3] To aby nezabili celú jeseň odhŕňaním lístia...

Úloha: Máme číslo N a súradnice $2N$ bodov. Zistite, či mohli tieto body vzniknúť posunutím vrcholov vhodného pravidelného $2N$–uholníka po polpriamkach od jeho stredu. Body nemusia byť zadané v žiadnom konkrétnom poradí.

Príklad:

Vstup:	**Výstup:**

$N = 2$
`6,3`
`6,9`
`3,5`
`11,5`
	Áno.
	
	Pôvodný štvorec mohol mať napr. vrcholy (6,6), (7,5), (6,4), (5,5).

4. O kanárikovi

15 bodov

Dávidko má veľmi rád kanáriky (napríklad na splave o nich dokáže spievať celé noci). Preto keď sa blížili jeho narodeniny, rodičia ani dlho nerozmýšľali, čo mu dať – dostal kanárika. Dávidko sa mu samozrejme veľmi potešil, urobil mu klietku, nakŕmil ho... Po čase ho však prestalo baviť počúvať stále to jednotvárne čvirikanie, a tak sa rozhodol, že ho naučí rozprávať. Ako správny programátor si naprogramoval generátor náhodných reťazcov, ktoré potom kanárika učil. Aby mal dôkaz pre kamarátov, nahrával si rozprávajúceho kanárika na CD. Čoskoro si všimol, že príliš dlhé reťazce si kanárik nedokáže zapamätať. Chcel zistiť, koľko najviac znakov kanárik zvládne, ale nejako sa pri tom zamotal – kanárik totiž pri nahrávaní opakoval reťazec stále dokola, a tak z nahratého CD nie je jasné, z koľkých písmen sa reťazec skladá. Nám už ale začal školský rok a Dávidko nejako nestíha, preto potrebuje vašu pomoc.

Úloha: Vašou úlohou je napísať program, ktorý z daného reťazca vypočíta dĺžku jeho najkratšej periódy. Perióda reťazca je také číslo p, že pre všetky i sú i-ty a $(i+p)$-ty znak reťazca rovnaké (ak existujú).

Príklad:

Vstup:	**Výstup:**
`cvirikcvirikcvirikcvirikcvir`	6

5. O mravcoch I

15 bodov

Mravce dostali na narodeniny kráľovnej počítače. Každý mravec dostal svoj vlastný maličký počitač. Samozrejme, okamžite ich zosieťovali. Keď ich prestalo baviť hrať sieťové hry, rozhodli sa, že si niečo naprogramujú. Programovať však vedel len Ferdo, preto kráľovná rozhodla, že Ferdo napíše program, ten nahrajú na niekoľko počítačov a naraz na všetkých ho spustia.

Vašou úlohou bude samozrejme pomáhať Ferdovi, preto si teraz bližšie popíšeme, ako bude vyzerať programovací jazyk, v ktorom sa programujú mravčie počítače. Program budeme písať v jazyku podobnom jazyku Pascal. Premenné použité v programe budú lokálne, t.j. každý počítač bude mať vlastné premenné. Jedinou výnimkou je globálne pole celých čísel `Mem[]`. Toto pole je indexované od 0 a neobmedzene veľké. Na začiatku je v ňom (väčšinou v prvých niekoľko políčkach) uložený vstup úlohy, ostatné jeho políčka obsahujú nuly. Na konci v ňom má ostať uložený výsledok úlohy.

Programy budú spustené naraz na všetkých počítačoch. Počítače budú očíslované od 1 do K. Výpočet bude prebiehať v taktoch. Všetky počítače sú rovnako rýchle, v každom takte každý z nich vykoná práve 1 inštrukciu.

Na začiatku programu má byť deklarácia lokálnych premenných, nasledujú samotné inštrukcie. V programe musí každá inštrukcia byť uvedená na samostatnom riadku (teda v jednom takte vykoná každý počítač práve jeden riadok programu). Riadok môžeme ukončiť bodkočiarkou, ňou zároveň začína komentár. Povolené inštrukcie sú:

- priradenie
- prázdna inštrukcia **nop**
- inštrukcia ukončenia výpočtu **halt**
- príkaz skoku **goto** *návestie*
- podmienený príkaz **if** *podmienka* **then** *inštrukcia* (**else** *inštrukcia*)

Pravú stranu priradenia môže predstavovať ľubovoľný aritmetický výraz, podmienkou môže byť ľubovoľný logický výraz. Okrem premenných a konštánt môžu tieto výrazy obsahovať volanie funkcie **cpuid**, ktorá vráti číslo počítača, na ktorom program beží. (Všimnite si, že nemáte k dispozícii kľúčové slová **begin** a **end**.) Návestia sa definujú rovnako ako v Pascale: napísaním *názov:* pred príslušný riadok.

Zostáva vyriešiť prácu so spoločnou pamäťou. Môže sa stať, že niekoľko počítačov chce v tom istom takte čítať z, resp. zapisovať na to isté pamäťové miesto. V takomto prípade najskôr prebehne čítanie, potom zápis, Čítať jedno políčko globálnej pamäte môže naraz ľubovoľne veľa počítačov, ale zapisovať dovolíme len jednému. Ak by naraz chcelo na to isté miesto zapisovať viac počítačov, program vráti chybu. (T.j. tomuto sa pri písaní programu treba vyhnúť.)

Ukážeme si teraz, v čom je sila mravčích počítačov. Na klasickom počítači potrebujeme na nájdenie najväčšieho z daných N čísel čas $O(N)$. Pozrime sa, ako rýchlo ho vedia nájsť mravce. Nech teda v `Mem[0]` je uložené N a v `Mem[1]` až `Mem[N]` jednotlivé čísla. Spustime na N počítačoch nasledovný program:

```
   var N,num1,num2 : integer;
   N:=Mem[0];
1: if (2*cpuid-1 > N) then halt;
   num1:=Mem[2*cpuid-1];
   if (2*cpuid <= N) then num2:=Mem[2*cpuid] else num2:=num1;
   if (num1<num2) then Mem[cpuid]:=num1 else Mem[cpuid]:=num2;
   N:=(N+1) div 2;
   if (N=1) then halt;
   goto 1
```

Po skončení bude v `Mem[1]` uložené najväčšie spomedzi daných čísel. Prečo? Počas prvého prechodu cyklom sme rozdelili zadané čísla na dvojice, z každej sme vybrali to väčšie a následne sme „víťazov“ uložili na políčka $1 \ldots \lceil N/2 \rceil$ – tým sme vlastne dostali pôvodnú úlohu s približne polovičným počtom čísel, to celé v konštantnom čase! Preto náš program naozaj nájde maximum spomedzi daných N čísel, a to v čase $O(\log N)$.

Úloha: Vašou úlohou bude napísať program, ktorý spočíta čiastočné súčty danej postupnosti. Teda nech v `Mem[0]` je uložené N a v `Mem[1]` až `Mem[N]` jednotlivé celé čísla. Po skončení by v `Mem[i]` (pre $1 \leq i \leq N$) mala byť uložená hodnota `Mem[1]+...+Mem[i]`. Môžete uviesť, koľko počítačov má byť na začiatku spustených v závislosti od N, počet počítačov však smie od N závisieť nanajvýš polynomiálne (teda $N^4 + 7$ počítačov je v poriadku, ale 2^N už nie.)

Prvoradým kritériom hodnotenia bude *čas výpočtu vášho programu*, nasleduje *použitá pamäť* (súčet lokálnych pamätí spustených počítačov a počtu použitých políčok globálnej pamäte) a *počet použitých počítačov*.