59. ročník Matematické olympiády – 2009/2010

Úlohy krajského kola kategorie P

Krajské kolo 59. ročníku MO kategorie P se koná v úterý 12. 1. 2010 v dopoledních hodinách. Na řešení úloh máte 4 hodiny čistého času. V krajském kole MO-P se neřeší žádná praktická úloha, pro zajištění rovných podmínek řešitelů ve všech krajích je použití počítačů při soutěži zakázáno. Zakázány jsou rovněž jakékoliv další pomůcky kromě psacích potřeb (tzn. knihy, výpisy programů, kalkulačky, mobilní telefony). Řešení každé úlohy vypracujte na samostatný list papíru.

Řešení každého příkladu musí obsahovat:

- **Popis řešení,** to znamená slovní popis principu zvoleného algoritmu, *argumenty zdůvodňující jeho správnost* (případně důkaz správnosti algoritmu), diskusi o efektivitě vašeho řešení (časová a paměťová složitost). Slovní popis řešení musí být jasný a srozumitelný i bez nahlédnutí do samotného zápisu algoritmu (do programu). Není vhodné odkazovat se na vaše řešení úloh domácího kola, opravovatelé je nemusí mít k dispozici; na autorská řešení se odkazovat můžete.
- **Zápis algoritmu.** V úlohách **P-II-1**, **P-II-2** a **P-II-3** je třeba uvést zápis algoritmu, a to buď ve tvaru zdrojového textu nejdůležitějších částí programu v jazyce Pascal nebo C/C++, nebo v nějakém dostatečně srozumitelném pseudokódu. Nemusíte detailně popisovat jednoduché operace jako vstupy, výstupy, implementaci jednoduchých matematických vztahů, vyhledávání v poli, třídění apod. V případě úlohy **P-II-4** je nutnou součástí řešení program pro počítač Kvak.

Za každou úlohu můžete získat maximálně 10 bodů. Hodnotí se nejen správnost řešení, ale také kvalita jeho popisu a efektivita zvoleného algoritmu.

Vzorová řešení úloh naleznete krátce po soutěži na webových stránkách olympiády *http://mo.mff.cuni.cz/*. Na stejném místě bude zveřejněn i seznam úspěšných řešitelů postupujících do ústředního kola a také popis prostředí, v němž budete v ústředním kole řešit praktické úlohy.

P-II-1 Aquapark

V aquaparku mají tři tobogany. Správce aquaparku se rozhodl zjistit, nakolik je návštěvníci využívají. Má k dispozici následující informace:

- Jak dlouho trvá jedna jízda na každém toboganu (časy T_1, T_2, T_3).
- Jak dlouho trvá, než návštěvník dojde od konců toboganů k jejich začátkům (čas D). Tobogany jsou umístněny tak, že cesta od konce libovolného toboganu k začátku libovolného toboganu trvá stejně dlouho.
- Pro každý tobogan fotobuňkou zaznamenané časy, kdy některý z návštěvníků aquaparku nasedl na tento tobogan.

Z těchto informací se přesný počet návštěvníků využívajících tobogany většinou nedá určit. Můžeme ale určit minimální počet lidí, kteří mohli tobogany využívat tak, aby to odpovídalo zaznamenaným údajům.

Soutěžní úloha

Na základě informací o délce jízdy na toboganech, trvání cesty zpět nahoru a časech jednotlivých nasednutí na tobogany určete minimální možný počet lidí, kteří mohli využívat tobogany.

Jinými slovy, najděte nejmenší číslo K takové, že existuje rozvrh pohybu pro K lidí, podle kterého někdo nasedne v každém ze zaznamenaných časů na příslušný tobogan. Nezapomeňte, že když někdo nasedne na tobogan i v čase T, tak další jízdu (na kterémkoliv z toboganů) může tento člověk začít nejdříve v čase $T + T_i + D$.

Nezapomeňte také zdůvodnit správnost svého algoritmu.

Formát vstupu

Na prvním řádku vstupu jsou tři čísla T_1, T_2, T_3 – délky jízd na jednotlivých toboganech. Na druhém řádku je jedno číslo D – čas potřebný na výstup od konců toboganů nahoru k jejich začátkům. Následují tři řádky s informacemi o uskutečněných jízdách na jednotlivých toboganech. Každý z nich začíná číslem N_i, udávajícím počet jízd, které se na toboganu i uskutečnily. Na řádku pak následuje N_i čísel a_{ij} $(1 \leq j \leq N_i)$, která představují časy nasednutí na tento tobogan. Pro každý tobogan je tato posloupnost časů uspořádána vzestupně.

Platí: $0 \leq N_1, N_2, N_3 \leq 1\,000\,000$, $1 \leq T_1, T_2, T_3, D, a_{ij} \leq 500\,000\,000$.

Formát výstupu

Vypište jediné číslo – minimální počet návštěvníků aquaparku, pro něž mohla zaznamenaná situace nastat.

Příklady

Vstup:	Výstup:

`1 2 3`
`1`
`2 1 7`
`3 2 5 11`
`1 3`
	`2`

Jeden možný způsob, jak mohli dva lidé uskutečnit všechny zaznamenané jízdy: první z nich jel na prvním, pak na třetím, a opět na prvním toboganu (v časech 1, 3 a 7), zatímco druhý člověk absolvoval všechny tři jízdy na druhém toboganu.

Vstup:	Výstup:

`4 5 6`
`10`
`2 2 3`
`3 1 7 15`
`1 5`
	`6`

V tomto případě žádný návštěvník nemohl stihnout dvě jízdy, proto jich určitě muselo být šest.

P-II-2 Oplocení farmy

Přemysl se doslechl, že se v zemědělství točí velké peníze, a rozhodl se, že v něm také začne podnikat. Netrvalo dlouho a už vlastnil rozlehlou farmu, na níž pěstoval množství zajímavých plodin. Přemyslova zemědělská půda má obdélníkový tvar a je rozdělena na $R \times S$ stejně velkých čtvercových políček. Na každém políčku se pěstuje jedna plodina.

Nedávno se farma ocitla v nebezpečí, neboť v jejím okolí se přemnožili zajíci, kteří si rádi pochutnávají na pěstovaných rostlinách. Proto se Přemysl rozhodl, že na farmě nechá postavit plot, který ochrání plody jeho práce.

Jelikož v okolí není velká konkurence v oblasti stavebnictví, Přemyslovi se podařilo sehnat pouze jeden kontakt – firmu Čtverce s.r.o., která se specializuje na stavební práce čtvercového charakteru. Firma Čtverce s.r.o. nabídla, že postaví na farmě plot, který ochrání zvolenou čtvercovou část pozemku.

Soutěžní úloha

Přemysl přemýšlí, kde má nechat plot postavit. Plot může chránit čtvercové území libovolné velikosti. Musí ale vést po hranicích mezi políčky, takže každé políčko ochrání buď celé, nebo vůbec. Přemysl navíc požaduje, aby plot chránil políčka s alespoň dvěma různými plodinami. Chce mít totiž jistotu, že nezůstane na trhu jenom s jedním produktem. Pomozte Přemyslovi zjistit, kolik má možností na postavení plotu.

Formát vstupu

Na prvním řádku vstupu jsou zadány rozměry pozemku – počet řádků R a počet sloupců S, a dále počet plodin K, které je možné na farmě pěstovat. Platí $1 \leq R, S \leq 2\ 500$, $1 \leq K \leq 1\ 000\ 000\ 000$. Na každém z dalších R řádků vstupu je vždy uvedeno S čísel – popis, které plodiny se pěstují na jednotlivých políčkách. Plodiny jsou označeny čísly od 0 do $K - 1$.

Formát výstupu

Vypište jediné číslo – kolika způsoby může Přemysl postavit na farmě plot, který ohradí čtvercovou oblast, na níž se pěstují aspoň dvě různé plodiny.

Příklad

Vstup:	Výstup:

`3 6 10`
`1 0 0 0 1 7`
`2 0 0 0 7 7`
`3 0 0 0 7 7`
	`8`

Máme pět možností, jak lze postavit plot kolem oblasti velikosti 2×2, *a tři možnosti, jak lze postavit plot kolem oblasti velikosti* 3×3.

P-II-3 Omezovač rychlosti

Společnost Expresní pošta doručuje zásilky po celé Evropě. V poslední době ale její řidiči často přehlíželi dopravní značky a dostávali pokuty za překročení maximální povolené rychlosti. Ředitel společnosti proto rozhodl, že nechá do každého auta namontovat omezovač rychlosti. Ten funguje následovně: řidič si na něm před jízdou nastaví maximální přípustnou rychlost v a přístroj se pak automaticky postará o to, aby auto během celé jízdy nikdy nepřekročilo tuto rychlost. Ředitel společnosti navíc vydal nařízení, že si řidič musí před každou jízdou nastavit takové omezení rychlosti, aby na trase, kterou pojede, nepřekročil žádnou maximální povolenou rychlost.

Soutěžní úloha

Je dána silniční síť, po níž jezdí řidiči společnosti Expresní pošta. Tato silniční síť obsahuje N měst, mezi nimiž vede celkem M různých cest. Každá cesta spojuje dvě města, přičemž cesty se mimo města kříží pouze mimoúrovňově (tzn. mimo města nelze odbočit z jedné cesty na jinou). Pro každou cestu známe její délku (v kilometrech) a maximální povolenou rychlost (v kilometrech za hodinu).

Pro danou dvojici měst x a y může existovat více způsobů, jak lze po cestách dojet z města x do města y. Napište program, který pro všechny dvojice měst x a y určí minimální čas potřebný na cestu z města x do města y při použití omezovače rychlosti a dodržení nařízení ředitele společnosti.

Formát vstupu

První řádek obsahuje dvě kladná celá čísla N, M ($1 \leq N \leq 50, 1 \leq M \leq 1\,000$) – počet měst a počet cest mezi nimi. Jednotlivá města jsou na vstupu označena čísly 1 až N.

Každý z následujících M řádků popisuje jednu cestu a obsahuje čtyři čísla i, j, d, m. Ta udávají, že cesta spojující města i a j má délku d kilometrů a maximální povolenou rychlost m kilometrů za hodinu. Všechny cesty jsou obousměrné. Mezi dvěma městy může být postaveno více cest různé délky a s různou maximální povolenou rychlostí.

Můžete předpokládat, že mezi každou dvojicí měst existuje aspoň jedna trasa (která může být tvořena více navazujícími cestami).

Všechny vzdálenosti a rychlosti jsou na vstupu uvedeny s přesností nejvýše na 3 desetinná místa. Pro každou vzdálenost d platí $1 \leq d \leq 1\,000\,000$, pro každou maximální povolenou rychlost m platí $5 \leq m \leq 100\,000$.

Formát výstupu

Výstup bude tvořen N řádky, z nichž každý obsahuje N čísel. Číslo v i-tém řádku a j-tém sloupci určuje minimální čas (v hodinách) potřebný na jízdu mezi městy i a j. Výsledek uveďte s přesností na tři desetinná místa.

Při práci s reálnými čísly v počítači mohou vznikat zaokrouhlovací chyby. Tuto skutečnost můžete ve svém řešení ignorovat – postupujte, jako kdyby všechny výpočty, které provádíte, byly přesné.

Příklad

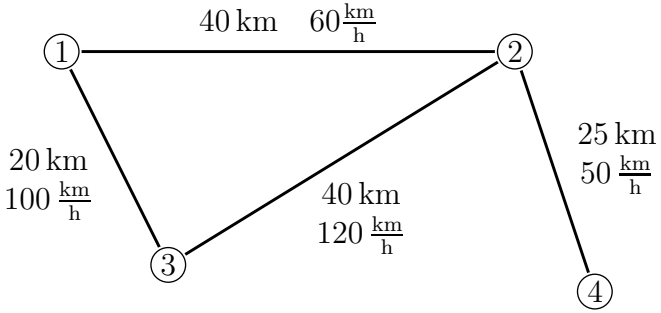
Vstup:

```
4 4
1 2 40.0 60.0
1 3 20.0 100.0
2 3 40.0 120.0
2 4 25.0 50.0
```

Výstup:

```
0.000 0.600 0.200 1.300
0.600 0.000 0.333 0.500
0.200 0.333 0.000 1.300
1.300 0.500 1.300 0.000
```

Z města 1 do města 2 se nejrychleji dostaneme přes město 3: pojedeme celkem 60 kilometrů rychlostí 100 km/h. Nejrychlejší cesta z města 1 do města 4 ovšem vede po trase 1-2-4, nikoliv 1-3-2-4.



P-II-4 Počítač Kvak

V tomto ročníku olympiády se setkáváme se speciálním počítačem nazvaným Kvak. Ve studijním textu uvedeným za zadáním úlohy je popsáno, jak tento počítač funguje. Studijní text je identický s textem z domácího kola.

Soutěžní úloha

a) *(3 body)*

Lucasova čísla jsou definována následovně: $L_0 = 2$, $L_1 = 1$ a pro každé $n \geq 2$ platí $L_n = L_{n-1} + L_{n-2}$.

V rouře počítače je jedno číslo n. Napište program pro počítač Kvak, který spočítá a vypíše hodnotu ($L_n \bmod 65\,536$).

b) *(3 body)*

V rouře počítače je neprázdná posloupnost *kladných* čísel. Napište program pro počítač Kvak, který zjistí, zda se v této posloupnosti vyskytuje číslo 47 a podle toho vypíše buď číslo 1 (pokud ano) nebo číslo 0 (pokud tam není).

c) *(4 body)*

V rouře počítače je neprázdná posloupnost *kladných* čísel. Napište program pro počítač Kvak, který na výstup vypíše všechna sudá čísla obsažená ve vstupní posloupnosti. Nezáleží přitom na pořadí, v jakém je vypíše, ale každé sudé číslo musí vypsat přesně tolikrát, kolikrát se vyskytlo na vstupu.

Studijní text

V letošním ročníku olympiády se budeme setkávat se speciálním počítačem zvaným Kvak.

Jediný datový typ, se kterým Kvak pracuje, se nazývá `number`, což je celé číslo z rozsahu od 0 do 65 535 včetně.* Všechny matematické výpočty provádí Kvak modulo 65 536, takže například hodnotou výrazu $65530 + 10$ je 4.

Kvak používá 26 proměnných, které nazýváme *registry.* Registry jsou označeny písmeny `a` až `z` a v každém z nich může být uložena jedna hodnota typu `number`. Na začátku výpočtu jsou ve všech registrech nuly.

Kromě registrů má Kvak ještě jednu *jednosměrnou rouru* neomezené délky, do které se mohu ukládat hodnoty typu `number`. Je to jediná datová struktura, kterou Kvak používá. S rourou lze provádět dvě operace:

- vložit do ní číslo z registru X příkazem `put` X,
- z opačného konce roury odebrat číslo a uložit ho do registru X příkazem `get` X.

Čísla se v rouře počítače nemohou předbíhat, Kvak je tedy bude odebírat ve stejném pořadí, v jakém je do roury vložil.** Roura má neomezenou kapacitu, lze do ní vložit libovolné množství čísel. Není-li řečeno jinak, roura je na začátku výpočtu prázdná.

Počítač Kvak má také možnost vypisovat čísla (výsledky výpočtu) na výstup.

Příkazy

V následující tabulce jsou shrnuty všechny příkazy, které Kvak umí provádět a které tedy můžete používat v programech.

příkaz	*význam příkazu*
`get` X	Kvak odebere jedno číslo z roury a uloží ho do registru X.
`put` X	Kvak vloží do roury číslo z registru X.
`put` *číslo*	Kvak vloží dané číslo do roury.
`print`	Kvak odebere jedno číslo z roury a vypíše ho na výstup.
`add`	sčítání: Kvak odebere dvě čísla z roury a vloží do roury jejich součet.
`sub`	odčítání: Kvak odebere dvě čísla z roury a vloží do roury jejich rozdíl (první minus druhé).
`mul`	násobení: Kvak odebere dvě čísla z roury a vloží do roury jejich součin.
`div`	dělení: Kvak odebere dvě čísla z roury a vloží do roury celou část jejich podílu (první lomeno druhé).

* $65\,535 = 2^{16} - 1$, typ `number` je tedy přesně to, co znáte jako 16-bitové celé číslo bez znaménka.

** Takovou datovou strukturu obvykle nazýváme *fronta.*

`mod`	zbytek: Kvak odebere dvě čísla z roury a vloží do roury zbytek, který dá první z nich po celočíselném dělení druhým.
`label` L	návěstí: Toto místo v programu dostane označení L (kde L může být libovolný řetězec). Stejné návěstí nesmí být v programu vícekrát.
`jump` L	skok: Kvak bude pokračovat v provádění programu od místa, které má označení L.
`jz` X L	skok jestliže nula: Je-li v registru X nula, Kvak provede příkaz `jump` L.
`jeq` X Y L	skok jestliže se rovnají: Je-li v registrech X a Y stejná hodnota, Kvak provede příkaz `jump` L.
`jgt` X Y L	skok jestliže je větší: Je-li v registru X větší hodnota než v registru Y, Kvak provede příkaz `jump` L.
`jempty` L	skok jestliže je prázdná: Není-li v rouře žádné číslo, Kvak provede příkaz `jump` L.
`stop`	konec: Kvak ukončí svůj výpočet.

Pokud se během výpočtu stane, že se pokusíme odebrat číslo z roury počítače a roura přitom bude prázdná, nastane chyba. Chyba nastane také tehdy, když se pokusíme dělit nulou, počítat zbytek po dělení nulou, nebo skočit na neexistující místo v programu. Dojde-li výpočet programu na konec, Kvak po provedení posledního příkazu korektně skončí (jako kdyby na konci programu byl ještě příkaz `stop`.)

V zápisu programu můžeme psát více příkazů na jeden řádek, v takovém případě je od sebe oddělujeme středníkem.

Příklad 1

Následující program spočítá a vypíše součet všech čísel od 1 do 20.

```
put 20
put 0
label start
get a
jz a end
put a ; put a ; put 1
add
sub
get b ; put b
jump start
label end
print
```

Pokaždé, když se Kvak při provádění programu dostane ke třetímu řádku (`label start`), budou v rouře právě dvě čísla. Jestliže první z nich označíme N, hodnota druhého bude rovna součtu $S = (N + 1) + \cdots + 20$. Poté načteme N do registru `a`. Je-li $N = 0$, máme v rouře hledaný součet, můžeme ho vypsat na výstup a skončit. V opačném případě chceme provést dvě věci: Přičíst N k dosud získanému

součtu, a následně N zmenšit o 1. Po provedení řádku šest (tři příkazy `put`) máme v rouře postupně čísla: S, N, N, 1. Příkaz `add` sečte první dvě, po jeho provedení bude v rouře trojice čísel N, 1, $N+S$. Po vykonání dalšího příkazu `sub` budou v rouře hodnoty $N+S$ a $N-1$. To už je téměř to, co potřebujeme, jenom v opačném pořadí. Proto první z nich načteme do registru `b` a znovu vložíme do roury.

Příklad 2

V rouře je neprázdná posloupnost čísel. Napíšeme program, který spočítá a vypíše na výstup jejich součet. (Přesněji, jeho zbytek po dělení 65 536.)

Budeme stále opakovat následující postup: Zjistíme, zda jsou v rouře aspoň dvě čísla. Jestliže ano, některá dvě z nich sečteme a nahradíme je jejich součtem. Pokud tam už dvě čísla nejsou, zůstalo tam tady už jenom jediné a to zjevně součtem všech původních čísel. V programu pro počítač Kvak můžeme tuto myšlenku implementovat například následovně:

```
label cyklus
get a
jempty konec
put a
add
jump cyklus

label konec
put a
print
```

Na začátku každé iterace odebereme z roury jedno číslo a vložíme ho do registru `a`. Pokud se tím roura vyprázdnila, máme v registru `a` hledaný součet, stačí ho už jenom vypsat. Pokud ne, číslo z registru `a` vrátíme zpět do roury. V tom okamžiku jsou v rouře alespoň dvě čísla a můžeme tedy bez obav provést příkaz `add`.

Časová složitost tohoto řešení je lineární vzhledem k počtu čísel, která byla na začátku výpočtu v rouře. Každá iterace cyklu totiž provádí jen konstantní počet příkazů a zmenší nám o jedno počet čísel v rouře.