59. ročník Matematické olympiády – 2009/2010

Úlohy domácího kola kategorie P

Úlohy P-I-1 a P-I-2 jsou prakticky zaměřené a vaším úkolem v nich je vytvořit a odladit efektivní program v jazyce Pascal, C nebo C++. Řešení těchto dvou úloh budete odevzdávat ve formě zdrojového kódu přes webové rozhraní přístupné na stránce *http://mo.mff.cuni.cz/submit/,* kde též naleznete další informace. Odevzdaná řešení budou automaticky vyhodnocena pomocí připravených vstupních dat a výsledky vyhodnocení se krátce po odevzdání dozvíte. Pokud váš program nezíská plný počet bodů, můžete své řešení opravit a znovu odevzdat.

Úlohy P-I-3 a P-I-4 jsou teoretické, vaším úkolem je nalézt efektivní algoritmus řešící zadaný problém. Řešení úlohy se tedy skládá z popisu navrženého algoritmu, zdůvodnění jeho správnosti (funkčnosti) a odhadu časové a paměťové složitosti. Součástí řešení je i zápis algoritmu ve formě zdrojového kódu nebo pseudokódu v úloze P-I-3 a v jazyce počítače Kvak v úloze P-I-4. Svá řešení můžete odevzdat ve formě souboru typu PDF přes výše uvedené webové rozhraní nebo zaslat poštou na adresu:

Matematická olympiáda – kategorie P
KSVI MFF UK
Malostranské náměstí 25
118 00 Praha 1

Řešení všech úloh můžete odevzdávat do 15. listopadu 2009. Opravená řešení úloh P-I-3 a P-I-4 dostanete zpět prostřednictvím krajských komisí MO. Seznam postupujících do krajského kola najdete na webových stránkách olympiády na adrese *http://mo.mff.cuni.cz/*, kde jsou také k dispozici další informace o kategorii P.

P-I-1 Malíř Bonifác

Radní v Kocourkově vypsali nedávno výběrové řízení na velmi odpovědnou a důležitou činnost: malování chodníku před radnicí. Ve výběrovém řízení zvítězil malíř Bonifác (jediný uchazeč a zcela náhodou také starostův bratr).

Jak už to bývá, sotva Bonifác podepsal smlouvu, hned začal dostávat z radnice jeden příkaz za druhým: Tento kus chodníku natřít zelenou barvou, tento růžovou, potom to skoro celé přetřít na bílo ... Netrvalo dlouho a Bonifác si všiml, že se některé příkazy překrývají. A když si uvědomil, že ho smlouva zavazuje provést všechny příkazy v tom pořadí, v jakém je dostal, začaly ho obcházet mdloby.

Naštěstí však přišel na geniální nápad. Kdyby věděl, jak má chodník vypadat nakonec po provedení všech příkazů, mohl by ho tak namalovat rovnou a potom se tvářit, že on přece všechny příkazy dodržel. A hlavně potom radnici všechno vyúčtuje podle původních příkazů a ještě na tom pořádně vydělá.

Soutěžní úloha

Chodník před radnicí měří K kocourkovských kroků. Jeden jeho konec bude mít souřadnici 0, opačný konec má souřadnici K. V současnosti má celý chodník

asfaltově černou barvu. Bonifác používá F jiných barev, očíslovaných od 1 do F. Postupně dostal N příkazů. Každý z nich zapíšeme ve tvaru 'a_i b_i f_i', kde a_i a b_i jsou souřadnice začátku a konce úseku a f_i je barva, kterou se má tento úsek obarvit. Na obarvení jednoho kocourkovského kroku chodníku potřebuje Bonifác jeden litr barvy. Pro každou z barev spočítejte, kolik litrů bude Bonifác potřebovat.

Formát vstupu

Vstupní soubor se jmenuje `bonifac.in`. Na prvním řádku souboru jsou tři celá čísla N (počet příkazů), F (počet barev) a K (délka chodníku) oddělená mezerami ($1 \leq N, F \leq 100\,000$, $1 \leq K \leq 1\,000\,000\,000$). Následuje N řádků, z nichž každý popisuje jeden příkaz, a to v pořadí, v jakém je Bonifác dostal. Přitom i-tý z těchto řádků obsahuje tři celá čísla a_i, b_i a f_i oddělená mezerami ($0 \leq a_i < b_i \leq K$, $1 \leq f_i \leq F$).

Pro 8 z 10 testovacích vstupů bude navíc platit $K \leq 1\,000\,000$. Pro 6 z těchto 8 testovacích vstupů bude navíc $N \leq 10\,000$, a pro 3 z těchto 6 vstupů bude $K, N \leq 1\,000$.

Formát výstupu

Výstupní soubor se jmenuje `bonifac.out`. Pro každou z F barev (v pořadí jejich čísel) zapište do výstupního souboru jeden řádek obsahující jedno celé číslo – kolik litrů této barvy bude Bonifác potřebovat.

Příklad

Vstupní soubor `bonifac.in`:	Výstupní soubor `bonifac.out`:
`4 5 7`	`1`
`1 5 1`	`3`
`2 4 3`	`1`
`4 6 4`	`0`
`3 6 2`	`0`

Odevzdávání řešení

Toto je praktická úloha. Odevzdáváte *pouze zdrojový kód odladěného programu* prostřednictvím webového rozhraní.

P-I-2 Čokoláda

Mařenka bude mít brzy narozeniny. Její bratr Jeníček dlouho nemohl vymyslet, co by jí jenom mohl k narozeninám dát – až konečně ve své tajné skrýši na půdě objevil zbytek čokolády, kterou si tam kdysi ukryl. Pravda, myši už si vybraly svoji daň, ale i tak z čokolády zůstalo ještě docela dost. Děravé části oláme, aby mu vznikla pěkná čtvercová tabulka, a tu úhledně zabalí. A zbytek samozřejmě sní.

Soutěžní úloha

Je dán původní počet řádků R a sloupců S, které čokoláda kdysi měla. Dále máme matici $R \times S$ nul a jedniček určující, která políčka čokolády zůstala celá. Zjistěte, kolika různými způsoby může Jeníček uskutečnit svůj plán. Jinými slovy řečeno,

spočítejte, kolika způsoby je možné ve zbytku čokolády vyznačit čtverec (libovolné velikosti) bez děr. Všechny hrany čtverce musí samozřejmě ležet na hranách políček. Stejně velké čtverce ležící na různých souřadnicích v tabulce čokolády považujeme za různá řešení.

Formát vstupu

Vstupní soubor se jmenuje `cokolada.in`. Na jeho prvním řádku jsou dvě celá celé čísla R a S oddělená mezerou ($1 \leq R, S \leq 2\,500$). Následuje R řádků, v r-tém z nich je S mezerami oddělených celých čísel $a_{r,1}, \ldots, a_{r,S}$. Je-li políčko čokolády (r, s) celé, je $a_{r,s} = 1$, jinak $a_{r,s} = 0$.

Pro 7 z 10 testovacích vstupů bude navíc platit $R \leq 500$. Pro 5 z těchto 7 testovacích vstupů bude $R, S \leq 500$, a pro 3 z těchto 5 vstupů bude $R, S \leq 20$.

Formát výstupu

Výstupní soubor `cokolada.out` obsahuje jediný řádek a na něm jedno celé číslo – hledaný počet čtverců.

Příklad

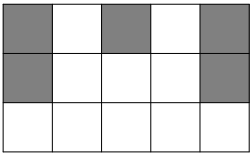
Vstupní soubor `cokolada.in`:

```
3 5
0 1 0 1 0
0 1 1 1 0
1 1 1 1 1
```

Výstupní soubor `cokolada.out`:

```
12
```

Na obrázku vpravo je nakreslena čokoláda popsaná ukázkovým vstupem. Šedou barvou jsou vyznačena políčka, která chybějí.



Čtverec 1×1 na ní můžeme vyznačit deseti způsoby a čtverec 2×2 dvěma, což je celkem $10 + 2 = 12$ způsobů.

Odevzdávání řešení

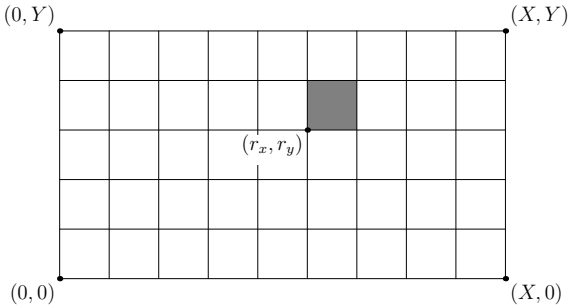
Toto je praktická úloha. Odevzdáváte *pouze zdrojový kód odladěného programu* prostřednictvím webového rozhraní.

P-I-3 Koláč

Zlá ježibaba drží v kleci Jeníčka a Mařenku a snaží se je vykrmit. Právě pro ně upekla plech ježibabího koláče. Koláč má tvar obdélníka, celý je odpudivý a navíc je ozdoben ohavnou pečenou ropuchou.

Protože cokoliv je lepší než muset sníst tuto ropuchu, rozhodli se Jeníček s Mařenkou, že si z jedení koláče udělají hru. Mařenka na něm lžičkou nakreslila čáry, čímž ho rozdělila na $X \times Y$ stejných čtverců. Celá ropucha sedí na jednom z těchto čtverců.

Jeníček s Mařenkou se nyní budou pravidelně střídat na tahu. Ten z nich, kdo je na tahu, si vybere některou z vyznačených čar a podél ní koláč rozřízne na dvě obdélníkové části. Následně sní tu část koláče, ve které není ropucha. Kdo bude na tahu v okamžiku, kdy už z koláče zbude pouze poslední čtverec s ropuchou, prohrál a musí ropuchu sníst. První tah provádí Mařenka.



Soutěžní úloha

Jsou dány rozměry koláče X, Y a souřadnice r_x, r_y levého dolního rohu čtverce, v němž je ropucha.

a) *(2 body)*

Rozhodněte, kdo zvítězí v situaci znázorněné na obrázku – tedy pro $(X, Y) = (9, 5)$ a $(r_x, r_y) = (5, 3)$ – a popište jednu možnou strategii, která mu zabezpečí výhru.

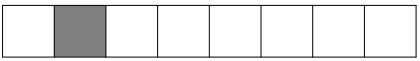
b) *(8 bodů)*

Popište co nejefektivnější algoritmus, který pro dané hodnoty X, Y, r_x a r_y zjistí, které z dětí hru vyhraje, jestliže budou obě hrát optimálně.

Příklady

Vstup:

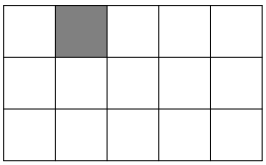
```
8 1 1 0
```



V prvním tahu Mařenka provede řez po přímce $x = 3$. Zbude ropucha a okolo ní z každé strany jeden čtverec. Jeníček sní jeden z nich, Mařenka druhý, a Jeníčkovi zůstane ropucha.

Vstup:

```
5 3 1 2
```



V druhém příkladu vyhraje Jeníček.

Odevzdávání řešení

Toto je teoretická úloha. Řešení odevzdejte ve formátu PDF prostřednictvím webového rozhraní, nebo ho zašlete poštou na adresu uvedenou v úvodu.

P-I-4 Počítač Kvak

V letošním ročníku olympiády se budeme setkávat se speciálním počítačem nazvaným Kvak. Ve studijním textu uvedeném za zadáním této úlohy je popsáno, jak počítač Kvak funguje a jak se programuje.

Soutěžní úloha

a) *(3 body)*

V rouře počítače je jedno číslo. Napište program pro Kvak, který vypíše 1, jestliže je to prvočíslo, zatímco v opačném případě vypíše 0.

Plný počet bodů dostanete za libovolné řešení, které bude mít méně než 100 příkazů a pro libovolný vstup vykoná méně než 10 000 kroků.

b) *(4 body)*

V rouře počítače je posloupnost *kladných* čísel. Délka této posloupnosti je menší než 65 000. Napište program pro Kvak, který tuto délku spočítá a vypíše. Plný počet bodů dostanete za řešení, které bude mít lineární časovou složitost.

c) *(3 body)*

Počítač Kvak se nám poškodil, takže dokáže provést příkaz `put` pouze desetkrát a poté se definitivně zastaví. Všechny ostatní příkazy provádí počítač bez problémů.

V rouře počítače je neprázdná posloupnost čísel. Je možné napsat program pro takto poškozený Kvak, který bez ohledu na délku vstupní posloupnosti spočítá a vypíše její maximum? Jestliže ano, napište takový program. V opačném případě dokažte, že to není možné.

Interpret

Na webové stránce olympiády budete mít k dispozici interpret programů pro počítač Kvak, abyste si mohli svoje řešení otestovat. (Interpret zveřejníme nejpozději měsíc před termínem odevzdání řešení domácího kola.)

Odevzdávání řešení

Toto je teoretická úloha. Řešení odevzdejte ve formátu PDF prostřednictvím webového rozhraní, nebo ho zašlete poštou na adresu uvedenou v úvodu.

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.

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

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).
`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.

** Takovou datovou strukturu obvykle nazýváme *fronta*.

`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.