Vojenská přehlídka

Termín odevzdání:	**10.12.2017 23:59:59**
Hodnocení:	**5.0000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	9 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je realizovat program, který bude počítat vozidla potřebná pro účast na vojenské přehlídce.

Předpokládáme, že organizujeme vojenskou přehlídku bojových vozidel. Vozidla na přehlídce jedou v útvaru, zabírají vždy celou šířku silnice. Tedy v každém pruhu jede jedno vozidlo, vozidla jedou stejně rychle vedle sebe. Pro estetický vzhled je potřeba, aby celá kolona vozidel měla obdélníkový tvar, tedy je potřeba, aby počet vozidel byl násobkem počtu pruhů silnice. Problém je, že silnice má v různých místech různý počet pruhů. V místech zúžení/rozšíření se kolona přeskupí tak, aby opět zabírala všechny pruhy silnice.

Vstupem programu je nejprve informace o počtu pruhů v jednotlivých úsecích silnice. Počty pruhů jsou celá kladná čísla, seznam je uveden ve složených závorkách, jednotlivé hodnoty jsou oddělené čárkami. Úsek ve složených závorkách je nejméně jeden, horní limit není stanoven. Po zadání úseků silnice jsou zadávané jednotlivé problémy. Každý problém je zadán jako dvojice čísel celých `from` a `to`. Číslo `from` označuje index prvního úseku, kterým kolona projede, indexy počítáme od 0. Index `to` označuje úsek, před kterým kolona ze silnice sjede.

Výstupem programu je nejmenší počet vozidel, aby kolona mohla projet všemi zadanými úseky silnice, zabírala vždy všechny pruhy a měla vždy obdélníkový tvar. Úloha má triviální řešení pro 0 vozidel, toto řešení je ale z propagandistických důvodů nepřípustné. Na druhou stranu má armáda jen velmi málo pojízdných vozidel, je tedy potřeba najít nejmenší možný nenulový počet vozidel, který vyhoví zadání.

Pokud je vstup neplatný, program to musí detekovat, zobrazit chybové hlášení a ukončit se. Chybové hlášení zobrazujte na standardní výstup (ne na chybový výstup). Za chybu považujte:

- počty pruhů silnice nebyla celá kladná čísla,
- nebyl zadán ani jeden úsek silnice,
- chybí nebo přebývají oddělovače v zadání úseků silnice (složené závorky, čárky),
- indexy `from` nebo `to` nejsou celá čísla,
- indexy `from` nebo `to` jsou mimo rozsah (překračují počet úseků silnice),
- index `from` je vyšší nebo roven indexu `to`.

Program je testován v omezeném prostředí. Je omezena doba běhu a dostupná paměť. Parametry jsou nastavené tak, aby jimi prošla správná implementace postavená na naivním algoritmu. Úloha vyžaduje, abyste si rozmysleli vhodnou reprezentaci úseků silnice v paměti. Dostupná paměť je omezena podle velikosti vstupů. Určitě neprojde řešení, které staticky alokuje paměť (např. 1000000 prvků). Úloha nabízí bonusový test. V tomto testu je zadaný velký počet úseků silnice a počítá se mnoho variant dotazů pro různé indexy `from` a `to`. Pro úspěšné zvládnutí bonusového testu je potřeba použít efektivní algoritmus a předzpracovat si vstupní data.

Ukázka práce programu:

```
Pocty pruhu:
{ 1, 2, 3, 4, 5, 6, 7, 8, 9 }
Trasy:
0 9
Vozidel: 2520
0 5
Vozidel: 60
0 6
Vozidel: 60
0 7
Vozidel: 420
0 8
Vozidel: 840
4 7
Vozidel: 210
2 9
Vozidel: 2520
8 9
Vozidel: 9
```

```
Pocty pruhu:
  {  7  , 3,5 ,11, 8, 16, 4, 9, 2
,8, 4, 2}
Trasy:
0 12
Vozidel: 55440
4 6
Vozidel: 16
4 7
Vozidel: 16
4 8
Vozidel: 144
  4    9
Vozidel: 144
4 10
Vozidel: 144
6 5
Nespravny vstup.
```

```
Pocty pruhu:
{ 1, 2, 3
5 6
Nespravny vstup.
```

```
Pocty pruhu:
{1,2,3}
Trasy:
2 4
Nespravny vstup.
```

```
Pocty pruhu:
{ 1, 2, trololo }
Nespravny vstup.
```

Nápověda:

- Ukázkové běhy zachycují očekávané výpisy Vašeho programu (tučné písmo) a vstupy zadané uživatelem (základní písmo). Zvýraznění tučným písmem je použité pouze zde na stránce zadání, aby byl výpis lépe čitelný. Váš program má za úkol pouze zobrazit text bez zvýrazňování (bez HTML markupu).
- Počty pruhů silnice na vstupu jsou celá čísla, která se vejdou do datového typu `int`.
- Výsledný počet vozidel může být velmi vysoký, je vhodné použít datový typ `long long int`.
- Slovní popis struktury platných vstupních dat není zcela exaktní. Proto připojujeme i formální popis vstupního jazyka v EBNF:

```
input      ::= { whiteSpace } '{' { whitespace } integer { lane } { whiteSpace } '}'
               { whiteSpace } { problem } { whiteSpace }
whiteSpace ::= ' ' | '\t' | '\n' | '\r'
lane       ::= { whiteSpace } ',' { whiteSpace } integer
problem    ::= { whiteSpace } integer { whiteSpace } integer
integer    ::= digit { digit }
digit      ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```