Kostky

Termín odevzdání:	**19.11.2017 23:59:59**
Hodnocení:	**3.9600**
Max. hodnocení:	**3.0000** (bez bonusů)
Odevzdaná řešení:	8 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	2 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je vytvořit program, který bude počítat počet různých kvádrů o zadaném objemu.

Předpokládáme kvádry, které mají hrany celočíselné délky. Zajímá nás, kolik existuje tvarově různých kvádrů o objemu nejméně `V1` a nejvíce `V2`. Za různé považujeme kvádry, které se liší ve svých rozměrech. Naopak, kvádry, které se liší pouze v otočení, považujeme za stejné. Tedy například kvádry 3x4x5, 4x3x5, 5x3x4, ... považujeme za stejné, ale kvádry 3x4x5 a 6x2x5 považujeme za různé.

Vstupem programu je seznam zadání. Každé zadání je jeden interval celých čísel. Interval udává minimální a maximální požadovaný objem kvádru. Pro každý interval program určí počet různých kvádrů v zadaném rozmezí objemů (takových kvádrů je konečný počet, protože uvažujeme pouze celočíselné délky hran).

Vstupní intervaly jsou zadané jako dvojice celých čísel v lomených závorkách oddělených středníkem. Takových intervalů může být na vstupu mnoho. Zadávání vstupních intervalů končí po ukončení standardního vstupu, tedy např. po stisku kláves Ctrl-D (Unix) nebo Ctrl-Z (Windows).

Výstupem programu je nalezený počet různých kvádrů pro zadané rozmezí objemů. Za každým výpisem je odřádkování.

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

- Rozmezí objemů ve vstupním intervalu není zadané jako dvojice celých čísel (nečíselná hodnota, chybí, ...),
- objem je nulový, záporný, dolní mez je vyšší než horní mez nebo horní mez překračuje 500000,
- chybějící nebo přebývající oddělovače (interval musí být zadaný v lomených závorkách, meze intervalu musí být oddělené středníkem).

Ukázka práce programu:

```
Intervaly:
<1;20>
Ruznych kvadru: 40
<15;45>
Ruznych kvadru: 90
```

```
Intervaly:
<15;15>
Ruznych kvadru: 2
<38;
75>
Ruznych kvadru: 132
   <   19  ;  120  ><20;25>
Ruznych kvadru: 374
Ruznych kvadru: 17
```

```
Intervaly:
<-5 ; 10>
Nespravny vstup.
```

```
Intervaly:
<2;6> < 4; foo> <9;11>
Ruznych kvadru: 7
Nespravny vstup.
```

Poznámky:

- 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).
- Znak odřádkování (\n) je i za poslední řádkou výstupu (i za případným chybovým hlášením).
- Pro reprezentaci objemů a rozměrů postačuje datový typ `int`.

- Úlohu lze vyřešit bez použití funkcí. Pokud ale správně použijete funkce, bude program přehlednější a bude se snáze ladit.
- Základní řešení úlohy vyžaduje použití cyklů a podmínek. Není potřeba používat pole. Rozhodně není dobrý nápad vygenerovat všechny možnosti a z nich odstraňovat duplicity - takové řešení by bylo těžkopádné, pomalé a pravděpodobně by selhalo na časovém nebo paměťovém limitu. V základní verzi je cílem získat zkušenosti s cykly s vhodně nastavenými mezemi.
- Úloha nabízí bonus, kterého lze dosáhnout, pokud je řešení časově efektivní. Časové efektivity lze dosáhnout zejména tím, že se neopakuje výpočet pro objem, který byl již dříve vyřešen. Pro získání bonusu je potřeba použít pole a několika dalších triků. Začínajícím programátorům doporučujeme řešit základní variantu bez bonusu (bez pole). I zkušeným programátorům doporučujeme nejprve zpracovat a odevzdat pouze základní jednodušší řešení.
- Pro načítání vstupu se hodí funkce `scanf`, podívejte se na konverze `"%c"` a `" %c"` (s mezerou před konverzí, najděte si v manuálu rozdíl).
- Při programování si dejte pozor na přesnou podobu výpisů. Výstup Vašeho programu kontroluje stroj, který požaduje přesnou shodu výstupů Vašeho programu s výstupy referenčními. Za chybu je považováno, pokud se výpis liší. I chybějící nebo přebývající mezera/odřádkování je považováno za chybu. Abyste tyto problémy rychle vyloučili, použijte přiložený archiv se sadou vstupních a očekávaných výstupních dat. Podívejte se na videotutoriál (edux -> výuková videa), jak testovací data použít a jak testování zautomatizovat.
- Slovní popis struktury platných vstupních dat není zcela exaktní. Proto připojujeme i formální popis vstupního jazyka v EBNF:

```
input      ::= { whiteSpace } { interval { whiteSpace } }
whiteSpace ::= ' ' | '\t' | '\n' | '\r'
interval   ::= '<' { whiteSpace } number { whiteSpace } ';' { whiteSpace } number { whiteSpace } '>'
number     ::= [ '+' ] digit { digit }
digit      ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```