Úkolem je realizovat program, který bude počítat pravoúhlé trojúhelníky.

Uvažujeme pravoúhlé trojúhelníky, které mají všechny strany celočíselné délky. Dále uvažujeme uzavřený interval celých čísel <a;b>. Chceme najít a spočítat/vypsat všechny pravoúhlé trojúhelníky, které mají celočíselnou délku stran a které mají všechny tři strany v zadaném intervalu hodnot. Například pro interval hodnot <4;14> nalezneme následující dva trojúhelníky:

```
5 12 13
6 8 10
```

Do odpovědi nepatří trojúhelníky 3 4 5 ani 9 12 15, protože některé jejich strany jsou mimo zadaný interval.

Při počítání trojúhelníků uvažujeme pouze unikátní tvary trojúhelníku. Tedy například trojúhelníky 5 12 13 a 6 8 10 jsou dva různé trojúhelníky, které oba zahrneme do výsledku. Naproti tomu trojúhelníky 5 12 13 a 12 5 13 jsou shodné, tedy do výsledku zahrneme pouze jeden z nich.

Vstupem programu je zadávání problémů k vyřešení. Program zadané problémy zpracovává a zobrazuje jejich řešení. Zpracování vstupu končí v okamžiku, kdy je zpracován poslední zadaný problém (je dosaženo EOF na standardním vstupu). Vstupní problémy jsou dvojího druhu:

- vypsání všech unikátních pravoúhlých trojúhelníků s celočíselnými délkami stran ze zadaného intervalu. Problém je zadán znakem otazník a mezemi intervalu:

  ```
  ? <lo;hi>
  ```

  kde lo a hi jsou celá čísla udávající meze intervalu. Odpovědí programu je seznam nalezených trojúhelníků (délek jejich stran), každý trojúhelník na jeden řádek (viz ukázka). Za koncem výpisu je zobrazen počet nalezených trojúhelníků,
- spočítání unikátních pravoúhlých trojúhelníků s celočíselnými délkami stran ze zadaného intervalu. Problém je zadán znakem hash a mezemi intervalu:

  ```
  # <lo;hi>
  ```

  kde lo a hi jsou celá čísla udávající meze intervalu. Odpovědí programu je počet nalezených trojúhelníků.

Pokud vstup není platný, program tuto situaci detekuje, vypíše chybové hlášení a ukončí se. Formát chybového hlášení je opět uveden v ukázkách níže. Za chybu je považováno:

- nerozpoznaný problém (nezačíná znakem ? ani #),
- nesprávné zadání mezí intervalu (meze musí být celá kladná čísla, dolní mez musí být menší nebo rovná horní mezi),
- chybějící oddělovače (menšítko, většítko, středník).

Pokud program detekuje chybu, přestane se dotazovat na další vstupní hodnoty, vypíše chybové hlášení a ukončí se. Chybu je tedy potřeba detekovat okamžitě po načtení hodnoty (neodkládejte kontrolu vstupních údajů až za načtení

celého vstupu). Chybové hlášení vypisujte na standardní výstup (nevypisujte jej na standardní chybový výstup).

Dodržte přesně formát všech výpisů. Výpis Vašeho programu musí přesně odpovídat ukázkám. Testování provádí stroj, který kontroluje výpis na přesnou shodu. Pokud se výpis Vašeho programu liší od referenčního výstupu, je Vaše odpověď považovaná za nesprávnou. Záleží i na mezerách, i na odřádkování. Nezapomeňte na odřádkování za posledním řádkem výstupu (a za případným chybovým hlášením). Využijte přiložený archiv s testovacími vstupy a usnadněte si testování Vašeho programu.

Váš program bude spouštěn v omezeném testovacím prostředí. Je omezen dobou běhu (limit je vidět v logu referenčního řešení) a dále je omezena i velikost dostupné paměti (toto by neměl být problém). Zadaný problém lze řešit algoritmy s různou efektivitou. Implementace opravdu velmi naivního algoritmu projde pouze povinnými testy, ale nezvládne časový limit testu nepovinného. Takové řešení bude hodnoceno méně než 100% bodů. Rozumný naivní algoritmus projde i testem nepovinným, bude hodnocen nominálními 100% bodů. Efektivní řešení je založené na algoritmu, který nemusí generovat všechny vyhovující trojúhelníky, tedy dokáže rychle spočítat výsledek i pro velké intervaly. Takové řešení projde i bonusovým testem.

Ukázka práce programu:

Problemy:
? <10;30>
*** 10 24 26**
*** 12 16 20**
*** 15 20 25**
*** 18 24 30**
*** 20 21 29**
Celkem: 5
? <8;60>
*** 8 15 17**
*** 9 12 15**
*** 9 40 41**
*** 10 24 26**
*** 12 16 20**
*** 12 35 37**
*** 14 48 50**
*** 15 20 25**
*** 15 36 39**
*** 16 30 34**
*** 18 24 30**
*** 20 21 29**
*** 20 48 52**
*** 21 28 35**
*** 24 32 40**
*** 24 45 51**
*** 27 36 45**
*** 28 45 53**
*** 30 40 50**
*** 33 44 55**
*** 36 48 60**
*** 40 42 58**
Celkem: 22
<8;60>

Celkem: 22
<7;120>
Celkem: 63

Problemy:
?<5;15>
*** 5 12 13**
*** 6 8 10**
*** 9 12 15**
Celkem: 3
< 3 ; 45 >
Celkem: 18
#<1;200>
Celkem: 127
<10;10>
Celkem: 0

Problemy:
* <3;9>
Nespravny vstup.

Problemy:
<12;11>
Nespravny vstup.

Problemy:
? 10;20
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).
- Výpisy trojúhelníků nelze příliš zoptimalizovat. Proto pro implementaci řešení problému typu výpis postačuje rozumná implementace naivního algoritmu. Efektivní algoritmus je potřeba pro implementaci problému typu spočtení trojúhelníků. Testy rychlosti zadávají pouze problémy typu

  ```
  # <lo;hi>
  ```

- Při počítání a vypisování trojúhelníků uvažujeme pouze unikátní tvary. Trojúhelníky, které se liší pouze pořadím stran (tj. jsou shodné), považujeme za identické a započteme/zobrazíme je pouze 1x. Například následující trojúhelníky jsou shodné:

  ```
  3 4 5
  4 3 5
  5 3 4
  ```

```
3 5 4
4 5 3
5 4 3
```

Z těchto trojúhelníků bude ve výpisu uveden pouze jeden (viz ještě poznámky níže) a do počtu trojúhelníků přidají 1.

- Podobné trojúhelníky, které nejsou shodné, zobrazíme/započteme každý zvlášť. Například trojúhelníky

  3 4 5 a

  6 8 10 jsou podobné, ale nejsou shodné. Tedy ve výpisu budou uvedené oba a do počtu přidají 2.

- Výpisy trojúhelníků nepředepisují pořadí vypisovaných trojúhelníků. Dále, vypisované trojúhelníky nemají určeno pořadí stran. Pořadí trojúhelníků ve výpisu a pořadí stran si Váš program může zvolit libovolně, testovací prostředí si pořadí před porovnáním upraví. Důležité je pouze do výpisu zahrnout všechny vyhovující trojúhelníky. Například pro ukázkové zadání ? <8;60> může být výpis:

```
* 8 15 17
* 9 12 15
* 9 40 41
* 10 24 26
* 12 16 20
* 12 35 37
* 14 48 50
* 15 20 25
* 15 36 39
* 16 30 34
* 18 24 30
* 20 21 29
* 20 48 52
* 21 28 35
* 24 32 40
* 24 45 51
* 27 36 45
* 28 45 53
* 30 40 50
* 33 44 55
* 36 48 60
* 40 42 58
Celkem: 22
```

nebo (prohození řádek):

```
* 9 12 15
* 8 15 17
* 9 40 41
* 10 24 26
* 12 16 20
* 12 35 37
* 14 48 50
* 15 20 25
* 15 36 39
```

* 16 30 34
* 18 24 30
* 20 21 29
* 20 48 52
* 21 28 35
* 24 32 40
* 24 45 51
* 27 36 45
* 28 45 53
* 30 40 50
* 33 44 55
* 36 48 60
* 40 42 58
Celkem: 22

nebo (prohození řádek a stran na řádce):

* 12 9 15
* 8 15 17
* 9 40 41
* 10 24 26
* 12 16 20
* 12 35 37
* 14 48 50
* 15 20 25
* 15 36 39
* 16 30 34
* 18 24 30
* 20 21 29
* 20 48 52
* 21 28 35
* 24 32 40
* 24 45 51
* 27 36 45
* 28 45 53
* 30 40 50
* 33 44 55
* 36 48 60
* 40 42 58
Celkem: 22

nebo v libovolné jiné ze zbývajících
1479428909100370019546403055652044799997 permutací.