Úkolem je vytvořit program, který bude počítat sekvence čísel.

Předpokládáme, že potřebujeme otevřít trezor. Trezor je zajištěn přístupovými kódy, každý kód je celé číslo v desítkové podobě. Pro zvýšení bezpečnosti je potřeba zadat jeden nebo více takových kódů. Při nastavování trezoru jsme si mohli vybrat, že přístupové kódy budou např. 3 a budou mít podobu např.: 151,0068,69. Náš trezor neomezuje počet kódů a neomezuje velikosti čísel, které kódy tvoří (čísla musí mít alespoň jednu cifru).

Problém je, že trezor je již nastavený, ale zadané kódy jsme zapomněli. Vedle trezoru máme pouze nalepený papírek se sekvencí čísel, která vznikla spojením všech zadaných kódů. Pro ukázku by to byla sekvence 151006869. Dále víme, že při vytváření kódů jsme dodrželi následující pravidla:

- pokud je kód liché číslo (např. 151 z ukázky), pak následující číslo může být libovolné,
- pokud je kód sudé číslo (např. 0068 z ukázky), pak následující číslo musí být stejně velké nebo větší.

Úkolem programu bude pro zadanou sekvenci číslic určit všechna možná rozdělení na kódy, které splňují podmínky výše. Dále program bude pro zadanou sekvenci číslic počítat, kolik takových rozdělení existuje.

Na vstupu jsou zadané jednotlivé problémy k vyřešení. Problémy jsou zadávané na řádkách, na každé řádce je zadaný právě jeden problém k vyřešení. Problémy jsou dvou druhů:

- pro zadanou posloupnost číslic určit a vypsat všechna možná rozdělení na kódy. Takový problém začíná znakem ?, za ním následuje nula, jeden nebo více bílých znaků a dále sekvence desítkových číslic k rozdělení. Výstupem je seznam všech možných rozdělení na kódy a dále celkový počet nalezených rozdělení (viz ukázka). Pořadí vypsaných řádek s kódy není určeno, může být libovolné.
- pro zadanou posloupnost číslic určit počet možných rozdělení na kódy. Zadání je podobné jako předešlé, jen začíná znakem #. Program nevypisuje jednotlivá nalezená rozdělení na kódy, zobrazí pouze celkový počet.

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:

- nerozpoznaný problém (prvním znakem na řádce není ani ? ani #),
- chybí posloupnost číslic k rozdělení na kódy,
- zadaná posloupnost obsahuje jiné znaky než desítkové číslice,
- po zadání posloupnosti číslic jsou na řádce ještě další znaky.

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. Řešený problém je poměrně komplikovaný z hlediska času potřebného pro výpočet. Očekává se základní řešení, které postupně zkusí všechny platné možnosti rozdělení vstupní sekvence. Takových rozdělení může být velmi mnoho, naivní řešení bude mít exponenciální složitost. Dále, pro delší vstupní sekvence lze snadno

překročit rozsah datového typu int, dokonce i typu long long. Proto je hodnocení rozděleno na několik částí:

- povinné testy zadávají krátké sekvence k rozdělení, postačuje rozumně efektivní implementace naivního algoritmu, který zkusí rozdělit vstup všemi přípustnými způsoby. Takové řešení neprojde nepovinným testem a dostane hodnocení menší než 100% bodů,
- nepovinný test rychlosti #1 zadává delší vstupní sekvence, kde při naivním řešení může dojít k překročení rozsahu datového typu long long. Vstupní sekvence jsou stále relativně krátké, je stále možné vyjít z naivního algoritmu. Je ale potřeba včas identifikovat neperspektivní rozdělení sekvence a neztrácet s nimi čas. Dále, nalezených rozdělení je v tomto testu mnoho, proto jsou v tomto testu zadávané pouze problémy typu # (výpis by příliš zdržoval). Program, který zvládne povinné testy i nepovinný test rychlosti #1, bude hodnocen nominálními 100% bodů,
- bonusový test zadává ještě delší vstupní sekvence, program nemá dost času na to, aby odzkoušel všechna možná rozdělení naivním algoritmem. Je potřeba identifikovat opakující se výpočty a eliminovat je. Protože nalezených rozdělení je velmi mnoho, jsou v tomto testu zadávané pouze problémy typu # (výpis by příliš zdržoval).

Ukázka práce programu:

Posloupnost:
12345
Celkem: 10
? 12345
*** 1,2,3,4,5**
*** 1,2,3,45**
*** 1,2,345**
*** 1,23,4,5**
*** 1,23,45**
*** 1,2345**
*** 12,345**
*** 123,4,5**
*** 123,45**
*** 12345**
Celkem: 10
#54321
Celkem: 6
? 54321
*** 5,4,321**
*** 5,43,21**
*** 5,4321**
*** 54,321**
*** 543,21**
*** 54321**
Celkem: 6

Posloupnost:
? 00000
*** 0,0,0,0,0**
*** 0,0,0,00**

*** 0,0,00,0**
*** 0,0,000**
*** 0,00,0,0**
*** 0,00,00**
*** 0,000,0**
*** 0,0000**
*** 00,0,0,0**
*** 00,0,00**
*** 00,00,0**
*** 00,000**
*** 000,0,0**
*** 000,00**
*** 0000,0**
*** 00000**
Celkem: 16
? 11111
*** 1,1,1,1,1**
*** 1,1,1,11**
*** 1,1,11,1**
*** 1,1,111**
*** 1,11,1,1**
*** 1,11,11**
*** 1,111,1**
*** 1,1111**
*** 11,1,1,1**
*** 11,1,11**
*** 11,11,1**
*** 11,111**
*** 111,1,1**
*** 111,11**
*** 1111,1**
*** 11111**
Celkem: 16
? 22222
*** 2,2,2,2,2**
*** 2,2,2,22**
*** 2,2,222**
*** 2,22,22**
*** 2,2222**
*** 22,222**
*** 22222**
Celkem: 7

Další ukázkové běhy naleznete v přiloženém archivu.

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).

- Zobrazení nalezených možných rozdělení vstupní sekvence (příkaz ?) nemá určené pořadí řádek ve výpisu. Pořadí může být libovolné, testovací prostředí si výpis před porovnáním upraví.
- Pokud budete převádět znaky ze vstupní sekvence na číslo, použijte datový typ long long. Tento typ může být potřeba i pro základní řešení.
- Počty nalezených řešení mohou být veliké. Pro bonusové testy nebude postačovat rozsah datového typu int, použijte long long.
- Nepovinný a bonusový test zadává dlouhé sekvence. Pokud byste vstupní sekvenci (nebo nějakou její větší část) převáděli na celé číslo, překročíte rozsah typu long long i nestandardního typu int128_t. Řešení vyžaduje nějakou vlastní implementaci čísel s neomezeným rozsahem. Tato implementace může být docela jednoduchá, úloha vyžaduje pouze několik málo operací, které s takovými čísly budete provádět. Implementace v referenčním řešení má cca 30 řádek kódu.
- Řešení této úlohy, které projde všemi závaznými a nepovinnými testy na 100%, může být použito pro code review.