Úkolem je realizovat program, který zoptimalizuje nákupy v supermarketech.

Nákupy obecně a zejména nákupy v supermarketu jsou pro některé potěšením (shopaholia) a pro některé černou noční můrou (shopafobia). Zatímco shopaholici s radostí projíždějí regály a zkoumají, které zboží je lepší či ve slevě (k velké radosti marketingových specialistů), shopafobici trpí při představě takto ztraceného času. Maximální očekávatelný výkon shopafobika je nákup podle předem připraveného seznamu. A zde se uplatní náš program.

Předpokládejme, že dostaneme nákupní seznam. Naším úkolem je nakoupit vyjmenované zboží a zároveň strávit v supermarketu co nejméně času. Nejvíce času se ztratí při vyhledávání zadaného zboží. Shopaholici čas při nakupování nezohledňují, proto uspořádání nákupního seznamu nevěnují řádnou péči. Při lineárním průchodu takového seznamu přejíždíme supermarketem tam a zpět, nakupování má pak časovou složitost až O(délka_seznamu * velikost_supermarketu). Vhodným přeuspořádáním seznamu lze dosáhnout časové úspory, kdy supermarketem projdeme pouze jednou a časovou složitost redukujeme na bolestivou, ale již snesitelnou hodnotu O(max(délka_seznamu, velikost_supermarketu)). To bude úkolem realizovaného programu.

Předpokládejme, že známe rozmístění zboží v regálech. Pro každý regál máme zadané jednotlivé druhy zboží (řetězce), každé zboží je na samostatné řádce. Regály jsou na vstupu zadané v pořadí 0, 1, 2, ..., toto číslování je zároveň pořadí, ve kterém procházíme supermarketem.

Po zadání regálů následuje prázdná řádka a pak jeden či více nákupních seznamů. Pokud je nákupních seznamů více, jsou oddělené prázdnou řádkou. Nákupní seznamy obsahují požadované zboží, opět každé zboží na samostatné řádce.

Program načte zadané rozmístění zboží do regálů a nákupní seznam (seznamy). Pro každý nákupní seznam pak vypíše jeho optimalizovanou podobu:

zboží na nákupním seznamu je uspořádáno tak, že je na regálech se vzrůstajícím (přesněji: neklesajícím) číslem,
pokud je zboží ve stejném regálu, tak vzájemné výsledné pořadí respektuje pořadí na původním seznamu,
pokud je zboží k dispozici na více regálech, bereme jej z prvního možného regálu,
pokud zboží není v supermarketu k dispozici, umístíme jej na konec seznamu.
Dalším drobným problémem je skutečnost, že nákupní seznamy nebývají přesné. Na seznamu se například vyskytuje "tomato", ale ve skutečnosti je v obchodě k dispozici "red tomato", "yellow tomato" a "Cherry Tomatoes in Box". Někteří v tom nevidí problém, je přece jasné že "k přípravě XXX je potřeba YYY." Náš program to bude řešit podobně jako průměrně inteligentní AI: pokud nenajdeme zboží stejného jména, vezmeme zboží, které obsahuje název požadovaného zboží jako podřetězec. Navíc (vzhledem k podmínkám výše) to znamená první vyhovující zboží z prvního možného regálu. Při porovnávání nebudeme rozlišovat malá a velká písmena.

Výstupem programu je upravený nákupní seznam s přeuspořádanými názvy zboží. Navíc pro každé zboží bude zobrazeno číslo regálu, kde se zboží nachází a název zboží, jak jej udává supermarket.

Program musí kontrolovat vstupní data. Pokud je vstup neplatný, program to musí detekovat, vypsat chybové hlášení a ukončit se. Za chybu je považováno:

chybějící číslo regálu při zadávání zboží (zadání regálu 0 musí být první řádka vstupu),
neplatné zadání čísla regálu,
čísla regálů netvoří sekvenci 0, 1, 2, ...,
chybí prázdná řádka za koncem zadávání zboží.
Pro zvládnutí základních a nepovinných testů postačuje rozumně implementovaný naivní algoritmus. Pro získání bonusu je potřeba implementovat algoritmus efektivnější, který zvládne dlouhé seznamy zboží.

Ukázka práce programu není tentokrát přímo na stránce zadání. Zadávané vstupy a očekávané výstupy jsou k dispozici v přiloženém archivu.

Nápověda:

Pro načítání řádek použijte funkci fgets. Nepoužívejte gets.
Vstupní řádky mohou být dost dlouhé, je potřeba s tím při návrhu programu počítat. Statická alokace nebude stačit.
Při řešení je vhodné použít datový typ struktura. Problém lze vyřešit i bez struktur, ale řešení se strukturami je přehlednější.