Úkolem je realizovat třídu `CInventory`, která bude implementovat databázi majetku (inventarizaci).

Naše databáze bude evidovat majetek ve firmě. Firma má pobočky a oddělení. Je potřeba evidovat, na kterém oddělení je jaký majetek umístěn. Majetek je popsán inventárním číslem (celé číslo, unikátní) a jménem (řetězec, může se opakovat). Oddělení je identifikované pomocí jména oddělení (např. accounting, human resources, ...), města a země. Jména oddělení, města i země se mohou opakovat. Jednoznačná identifikace oddělení je daná celou trojicí jméno `oddělení+město+země`. Předpokládáme, že majetek je vždy evidován právě na jednom oddělení.

Veřejné rozhraní je uvedeno níže. Pro třídu `CInventory` obsahuje následující:

- Konstruktor bez parametrů. Tento konstruktor inicializuje instanci třídy tak, že vzniklá instance je zatím prázdná (neobsahuje žádné záznamy).
- Destruktor. Uvolňuje prostředky, které instance alokovala.
- Metoda `AddProp(invID, name, dept, city, country)` přidá do existující databáze další záznam. Parametry `dept`, `city` a `country` reprezentují odddělení, `invID` je identifikace majetku a `name` je popis majetku. Metoda vrací hodnotu `true`, pokud byl záznam přidán, nebo hodnotu `false`, pokud dojde k chybě a majetek přidán nebyl. Chybou je, pokud již v databázi existuje majetek se stejným `invID`.
- Metoda `DelProp (invID)` odstraní záznam z databáze. Parametrem je jednoznačná identifikace majetku pomocí inventárního čísla. Pokud byl záznam skutečně odstraněn, vrátí metoda hodnotu `true`. Pokud záznam neodstraní (protože neexistoval majetek s touto identifikací), vrátí metoda hodnotu `false`.
- Metoda `Transfer ( invID, dept, city, country )` provede převod zadaného majetku (`invID`) ze stávajícího oddělení na nové oddělení, které je zadaného parametry `dept`, `city` a `country`. Pokud metoda uspěje, vrací `true`, pro neúspěch vrací `false` (neexistující majetek, chybou je i pokus o převod, kdy stávající a nové oddělení je to samé).
- Metoda `PropList` vyhledá majetek, který je evidován na daném oddělení. Metoda vrátí objekt typu `CPropList`, pomocí kterého lze postupně projít nalezené záznamy (iterátor). Pokud zadané oddělení neexistuje, je vrácen iterátor s prázdným obsahem (0 kusů majetku).
- Metoda `PropCount` zjistí počet položek majetku, který je evidovaný na zadaném oddělení. Pokud zadané oddělení v databázi neexistuje, je vrácena 0.
- Do třídy si budete muset přidat členské proměnné. Dále si do rozhraní můžete přidat další pomocné metody (ideálně neveřejné).

Třída `CPropList` slouží k jednorázovému procházení seznamem majetku, který je evidovaný na nějakém oddělení. V seznamu je vždy jedna položka vybraná a v seznamu se můžeme pohybovat vpřed. Rozhraní třídy `CPropList`:

- metoda `InvID` zjistí inventární číslo aktivní položky,
- metoda `Name` zjistí jméno aktivní položky,
- metoda `Remains` zjistí, zda (a kolik) položek majetku ještě zbývá projít,
- metoda `Next` posouvá v seznamu o jednu pozici směrem ke konci.

Odevzdávejte soubor, který obsahuje implementované třídy `CInventory` a `CPropList`. Třídy musí splňovat veřejné rozhraní podle ukázky - pokud Vámi odevzdané řešení nebude obsahovat popsané rozhraní, dojde k chybě při kompilaci. Do třídy si ale můžete doplnit další metody (veřejné nebo i privátní) a členské proměnné. Odevzdávaný soubor musí obsahovat jak deklaraci třídy (popis rozhraní) tak i definice metod, konstruktoru a destruktoru. Je jedno, zda jsou metody implementované inline nebo odděleně. Odevzdávaný soubor nesmí obsahovat vkládání hlavičkových souborů a funkci `main` (funkce `main` a vkládání hlavičkových souborů může zůstat, ale pouze obalené direktivami podmíněného překladu jako v ukázce níže).

Třída je testovaná v omezeném prostředí, kde je limitovaná dostupná paměť (dostačuje k uložení seznamu) a je omezena dobou běhu. Implementace třídy `CInventory` se nemusí zabývat kopírujícím konstruktorem ani přetěžováním operátoru =. V této úloze ProgTest neprovádí testy této funkčnosti.

Implementace třídy musí být efektivní z hlediska nároků na čas i nároků na paměť. Jednoduché lineární řešení nestačí (pro testovací data vyžaduje čas přes 5 minut). Předpokládejte, že operace převodů majetku, zjišťování počtů a zjišťování evidovaného majetku v zadaném oddělení je častější než operace vkládaní a mazání. Dále předpokládáme, že nová oddělení nevznikají příliš často (častější je přidání/převod majetku do již existujícího oddělení).

Pro ukládání záznamů v databázi alokujte prostor dynamicky, případně použijte kontejnery z STL. Pozor, pokud budete pole alokovat ve vlastní režii, zvolte počáteční velikost malou (např. tisíc prvků) a velikost zvětšujte/zmenšujte podle potřeby. Při zaplnění pole není vhodné alokovat nové pole větší pouze o jednu hodnotu, takový postup má obrovskou režii na kopírování obsahu. Je rozumné pole rozšiřovat s krokem řádově tisíců prvků, nebo geometrickou řadou s kvocientem ~1.5 až 2.

Pokud budete používat STL, nemusíte se starat o problémy s alokací pole. Z STL máte v této úloze k dispozici povolený kontejner `vector` (úmyslně ne `list`, `set`, `map` a další).

Požadované veřejné rozhraní třídy a ukázku použití naleznete v přiloženém archivu.