Kontrolní hlášení

Termín odevzdání:	**03.04.2022 23:59:59**	603689.204 sec
Pozdní odevzdání s penalizací:	**15.05.2022 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)	
Hodnocení:	**0.0000**	
Max. hodnocení:	**5.0000** (bez bonusů)	
Odevzdaná řešení:	0 / 20 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)	
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)	

Úkolem je realizovat třídu `CVATRegister`, která bude implementovat databázi kontrolních hlášení DPH.

Pro plánované důslednější potírání daňových úniků je zaveden systém kontrolních hlášení. V databázi jsou zavedené jednotlivé firmy, a do databáze jsou zaznamenávané jednotlivé vydané faktury, které daná firma vydala. Firmy lze do databáze zadávat a lze je rušit. Firma je identifikována svým jménem, adresou a daňovým identifikátorem (id). Daňový identifikátor je unikátní přes celou databázi. Jména a adresy se mohou opakovat, ale dvojice (jméno, adresa) je opět v databázi unikátní. Tedy v databázi může být mnoho firem `ACME`, mnoho firem může mít adresu `Praha`, ale firma `ACME` bydlící sídlící ve městě `Praha` může být v databázi pouze jedna. Při porovnávání daňových identifikátorů **rozlišujeme** malá a velká písmena, u jmen a adres naopak **nerozlišujeme** malá a velká písmena.

Veřejné rozhraní je uvedeno níže. 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 `newCompany(name, addr, id)` přidá do existující databáze další záznam. Parametry `name` a `addr` reprezentují jméno a adresu, parametr `id` udává daňový identifikátor. Metoda vrací hodnotu `true`, pokud byl záznam přidán, nebo hodnotu `false`, pokud přidán nebyl (protože již v databázi existoval záznam se stejným jménem a adresou, nebo záznam se stejným id).
- Metody `cancelCompany (name, addr) / cancelCompany (id)` odstraní záznam z databáze. Parametrem je jednoznačná identifikace pomocí jména a adresy (první varianta) nebo pomocí daňového identifikátoru (druhá varianta). Pokud byl záznam skutečně odstraněn, vrátí metoda hodnotu `true`. Pokud záznam neodstraní (protože neexistovala firma s touto identifikací), vrátí metoda hodnotu `false`.
- Metody `invoice (name, addr, amount) / invoice (id, amount)` zaznamenají příjem ve výši `amount`. Varianty jsou dvě - firma je buď identifikována svým jménem a adresou, nebo daňovým identifikátorem. Pokud metoda uspěje, vrací `true`, pro neúspěch vrací `false` (neexistující firma).
- Metoda `audit ( name, addr, sum ) / audit ( id, sum )` vyhledá součet příjmů pro firmu se zadaným jménem a adresou nebo firmu zadanou daňovým identifikátorem. Nalezený součet uloží do výstupního parametru `sum`. Metoda vrátí `true` pro úspěch, `false` pro selhání (neexistující firma).
- Metoda `medianInvoice ()` vyhledá medián hodnoty faktury. Do vypočteného mediánu se započtou všechny úspěšně zpracované faktury zadané voláním `invoice`. Tedy **nezapočítávají** se faktury, které nešlo přiřadit (volání `invoice` selhalo), ale započítávají se všechny dosud registrované faktury, tedy při výmazu firmy se **neodstraňují** její faktury z výpočtu mediánu. Pokud je v systému zadaný sudý počet faktur, vezme se vyšší ze dvou prostředních hodnot. Pokud systém zatím nezpracoval žádnou fakturu, bude vrácena hodnota 0.
- Metody `firstCompany ( name, addr ) / nextCompany ( name, addr )` slouží k procházení existujícího seznamu firem v naší databázi. Firmy jsou procházené v abecedním pořadí podle jejich jména. Pokud mají dvě firmy stejná jména, rozhoduje o pořadí jejich adresa. Metoda `firstCompany` nalezne první firmu. Pokud je seznam firem prázdný, vrátí metoda hodnotu `false`. V opačném případě vrátí metoda hodnotu `true` a vyplní výstupní parametry `name` a `addr`. Metoda `nextCompany` funguje obdobně, nalezne další firmu, která v seznamu následuje za firmou určenou parametry. Pokud za `name` a `addr` již v seznamu není další firma, metoda vrací hodnotu `false`. V opačném případě metoda vrátí `true` a přepíše parametry `name` a `addr` jménem a adresou následující firmy.

Odevzdávejte soubor, který obsahuje implementovanou třídu `CVATRegister`. Třída 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 kromě implementace třídy `CVATRegister` obsahovat nic jiného, zejména ne vkládání hlavičkových souborů a funkci `main` (funkce `main` a vkládání hlavičkových souborů může zůstat, ale pouze obalené direktivami podmíněného překladu). Za základ implementace použijte přiložený zdrojový soubor.

Třída je testovaná v omezeném prostředí, kde je limitovaná dostupná paměť (dostačuje k uložení seznamu) a je omezena dobou běhu. Implementovaná třída 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 vytvoření a likvidace firmy jsou řádově méně časté než ostatní operace, tedy zde je lineární složitost akceptovatelná. Častá jsou volání `invoice` a `audit`, jejich časová složitost musí být lepší než lineární (např. logaritmická nebo amortizovaná konstantní). Dále, v povinných testech se metoda `medianInvoice` volá málo často, tedy nemusí být příliš efektivní (pro úspěch v povinných testech stačí složitost lineární nebo `n log n`, pro bonusový test je potřeba složitost lepší než lineární).

Pro uložení hodnot alokujte pole dynamicky případně použijte 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. Pozor - k dispozici máte pouze část STL (viz hlavičkové soubory v přiložené ukázce). Tedy například kontejnery `map` / `unordered_map` / `set` / `unordered_set` / ... nejsou k dispozici.

V přiloženém zdrojovém kódu jsou obsažené základní testy. Tyto testy zdaleka nepokrývají všechny situace, pro odladění třídy je budete muset rozšířit. Upozorňujeme, že testy obsažené v odevzdaných zdrojových kódech považujeme za nedílnou součást Vašeho řešení. Pokud v odevzdaném řešení necháte cizí testy, může být práce vyhodnocena jako opsaná.

Poznámky:

- Pokud Vaše řešení projde všemi povinnými a nepovinnými testy na 100%, lze jej použít pro code review (tedy pro code review není nutné zvládnout poslední bonusový test). Protože se ale jedná o úlohu snazší a kratší, počítejte v případném code review s přísnějším hodnocením.

Vzorová data:		**Download**
Odevzdat:	Choose file	**Odevzdat**

☐ **Referenční řešení**