Kontrola kontrolního hlášení

Termín odevzdání:	**23.04.2017 23:59:59**	862591.255 sec
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 třídy, které implementují správu daně z přidané hodnoty.

Předpokládáme, že firmy si předávají faktury. Firma, která vydá fakturu, musí z fakturované částky odvést státu DPH. Analogicky, firma, která přijme (a zaplatí fakturu) může po státu nárokovat vratku DPH (za určitých okolností). Právě vracení DPH je slabým místem, které lze využít ke krácení daně a daňovým podvodům. Proto je úkolem implementovat jádro systému, který bude takové krácení daně ztěžovat.

Vydané a přijaté faktury eviduje třída `CVATRegister`. Firmy musí registrovat vydávané a přijímané faktury. Registr páruje dvojice sobě odpovídajících faktur a je schopen najít faktury nespárované. Fakturu lze do registru jednak přidat a dále i zrušit (např. při chybně zadaných údajích). Rozhraní třídy je následující:

implicitní konstruktor
: inicializuje prázdnou instanci registru,

RegisterCompany (name)
: metoda zavede zadanou firmu do registru. Předané jméno je oficiální název firmy, toto jméno bude používané v exportech z registru. Návratovou hodnotou je indikátor úspěchu (`true`)/neúspěchu (`false`). Za neúspěch považujte, pokud v registru již existuje firma stejného jména. Při porovnávání jména firmy je registr docela tolerantní:

  - při porovnávání nerozlišuje malá a velká písmena,
  - při porovnávání neuvažuje nadbytečné mezery.

  Tato pravidla jsou používána při zakládání nové firmy i vkládání / mazání faktur. Například názvy "My Company", "mY CoMpAnY", " My Company " a " mY CoMPAnY " jsou považované za jednu firmu, ale názvy "My Company" a "MyCompany" označují dvě rozdílné firmy.

AddIssued (invoice)
: metoda přidá fakturu do registru. Tuto metodu volá firma, která fakturu vydala. Návratovou hodnotou je příznak úspěch (`true`)/neúspěch (`false`). Za chybu je považováno, pokud prodávající / kupující ve faktuře nejsou registrované, prodávající a kupující je ta samá firma nebo pokud stejná faktura již byla pomocí `AddIssued` zadaná (dvě faktury se musí lišit alespoň v jednom z: prodávající/kupující/datum/částka/DPH).

AddAccepted (invoice)
: metoda přidá fakturu do registru, tuto metodu volá firma, která fakturu přijala (kupující). Jinak se metoda chová stejně jako `AddIssued`.

DelIssued (invoice)
: metoda odebere fakturu z registru. Tuto metodu volá firma, která fakturu vydala a dříve zaregistrovala. Návratovou hodnotou je příznak úspěch (`true`)/neúspěch (`false`). Za chybu je považováno, pokud identická faktura nebyla dříve registrovaná metodou `AddIssued`.

DelAccepted (invoice)
: metoda odebere fakturu z registru. Tuto metodu volá firma, která fakturu přijala a dříve zaregistrovala. Návratovou hodnotou je příznak úspěch (`true`)/neúspěch (`false`). Za chybu je považováno, pokud identická faktura nebyla dříve registrovaná metodou `AddAccepted`.

Unmatched (company, sortOpt)
: metoda nalezne všechny faktury, které se týkají zadané firmy `company` a nebyly spárované (tedy byly registrované pouze pomocí AddIssued nebo pouze pomocí AddAccepted). Metoda vrátí seznam těchto faktur, faktury budou seřazené podle kritérií udaných `sortOpt`. Faktury vrácené touto metodou budou mít na místě názvu firmy "oficiální" název, tedy ten název, který byl zadán při registraci firmy metodou `RegisterCompany`. Tento oficiální název bude rovněž použit při řazení.

Třída `CInvoice` reprezentuje jednu fakturu. Rozhraní musí splňovat:

konstruktor (date, seller, buyer, amount, vat)
: inicializace fakturu datem, jménem prodávající a kupující firmy, fakturovanou částkou a DPH.

Date, Seller, Buyer, Amount, VAT
: přístupové metody ke čtení jednotlivých složek faktury.

další
: do rozhraní faktury si můžete doplnit další veřejné/neveřejné metody a členské proměnné, které pro implementaci budete potřebovat.

Třída `CSortOpt` určuje kritéria pro řazení. Pro řazení lze použít všechny složky faktury. Pokud například vytvoříme instanci:

```
CSortOpt () . AddKey ( CSortOpt::BY_AMOUNT, true ) . AddKey ( CSortOpt::BY_SELLER, false )
```

pak se řadí podle fakturované částky vzestupně (první řadicí kritérium) a pro stejné hodnoty fakturované částky se použije řazení podle jména prodávajícího sestupně (druhé řadicí kritérium). Pokud by ani takto nebylo pořadí jednoznačně určené, použije se jako řadicí kritérium pořadí zavedení faktury do registru. Rozhraní třídy `CSortOpt` je:

implicitní konstruktor
: inicializuje prázdnou instanci třídy

AddKey (sortBy, ascending)
: přidá další řadicí kritérium `sortBy`, směr řazení je daný příznakem `ascending` (`true` = vzestupně, `false` = sestupně). Řadit lze podle:

- `BY_DATE` - podle data faktury,
- `BY_SELLER` - podle oficiálního jména prodávající firmy, řadí se bez ohledu na malá/velká písmena,
- `BY_BUYER` - podle oficiálního jména kupující firmy, řadí se bez ohledu na malá/velká písmena,
- `BY_AMOUNT` - podle fakturované částky,
- `BY_VAT` - podle DPH

Třída `CDate` implementuje jednoduché datum, její implementace je hotová v testovacím prostředí a pro testování ji máte dodanou v přiloženém archivu. Její implementaci nelze měnit, dodaná implementace musí zůstat v bloku podmíněného překladu.

Odevzdávejte zdrojový kód s implementací tříd `CVATRegister`, `CInvoice` a `CSortOpt`. Za základ implementace použijte přiložený soubor s deklarací metod a se sadou základních testů. Pro implementaci se může hodit doplnit i další pomocné třídy.

Zadání se trochu podobá dřívější domácí úloze. Předpokládáme, že při implementaci použijete vhodné kontejnery z STL (je k dispozici téměř celá), dále předpokládáme, že Vaše implementace bude časově a paměťově efektivní. Veškeré vkládání a mazání by mělo být rychlejší než lineární, řazení pak `n log n`. Není rozumné na všechny vnitřní struktury používat kolekci `vector`. Pokud chcete využívat C++11 kontejnery `unordered_set / unordered_map`, pak hashovací funktor neodvozujte jako specializaci `std::hash`. Hashovací funkci/funktor deklarujte explicitně při vytváření instance `unordered_set / unordered_map`. (Specializace `std::hash` předpokládá opětovné otevření jmenného prostoru `std`. To se těžko realizuje, pokud jste uzavřeni do jiného jmenného prostoru. Návody dostupné na internetu (stack overflow, cpp reference) implicitně předpokládají, že jmenné prostory nepoužíváte, na nich doporučovaná řešení nejsou ideálně kompatibilní.)

Vzorová data: Download

Odevzdat: Vybrat soubor Soubor nevybrán Odevzdat

Referenční řešení

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test s parametry podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 10.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty (bez vymazu)': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.443 s (limit: 10.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.516 s (limit: 9.557 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty + mem dbg': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.504 s (limit: 4.000 s)
    - Úspěch v nepovinném testu, hodnocení: 100.00 %
  - Test 'Test rychlosti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 25.00 %
    - Celková doba běhu: 5.412 s (limit: 15.000 s)
    - Úspěch v nepovinném testu, hodnocení: 100.00 %
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 1.00 * (5.00 + 0.50) = 5.50

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce

Funkce:	**20**	--	--	--
Řádek kódu:	**104**	**5.20 ± 3.60**	**16**	`LessBSD(const AInvoice &,const AInvoice &)`
Cyklomatická složitost:	**24**	**1.20 ± 1.33**	**5**	`LessBSD(const AInvoice &,const AInvoice &)`