OOP Midterm

> This midterm content were created by 黃漢軒 (109590031).

Folder Structure Tree

While your project has been built by `makefile`, the structure tree should be the same as the following section.

```
bin/
├─ ut_all
src/
├─ Product.h
├─ ShoppingCart.h
test/
├─ <Place_your_unit_test_file_here>
├─ ut_main.cpp
makefile
```

Problem Content

In this strange country, the people will buy only one product of the same type, so they don't buy the product more than one.

Because if they do, they will fail the OOP midterm. :(

In this task, you need to build a shopping cart.

- You should complete the `ShoppingCart.h` header file, store products, calculate the total cost, and support some operations of shopping cart.
- You should complete the `Product.h` header file, store the info of product.

A shopping cart should have items and support these operations:

- Union the item in Target Shopping Cart to the Source Shopping Cart by using plus operator overloading.
- Difference the items in Target Shopping Cart with the Source Sopping Cart by using minus operator overloading.

Formally to said, if the item set A in Source Shoping Cart, and the item set B in Target Shopping Cart

- The plus operator overloading should union two ordered-set of item, that is: $A \cup B$.
- The minus operator overloading should difference two ordered-set of item, that is $A - B$.

For example, if we have two carts, the first cart contains the item $\{B, C, A, D\}$, and the second cart contains $\{B, D, E\}$:

- When we doing the Union, the cart after we union will be $\{B, C, A, D, E\}$.
    - You should check the item is exist or not exist, to decide append or not append the item to the first cart, for example:
        - The first step we add first item B in the secord cart, but B already exist in the first cart, skip it.

          The current result should be $\{B, C, A, D\}$
        - The second step we add second item D in the secord cart, but D already exist in the first cart, skip it.

          The current result should be $\{B, C, A, D\}$
        - The third step we add third item E in the secord cart, E doesn't exist in the first cart, so append it to the item list of the first cart, the current result should be $\{B, C, A, D, E\}$
        - After these step, the result of union should be $\{B, C, A, D, E\}$.

- When we doing the Difference, the cart after we difference will be $\{C, A\}$.
    - You should check the item is exist or not exist, to decide delete or not delete the item to the first cart, for example:
        - The first step we delete first item B in the second cart, B exist in the first cart, so we delete it, $\{C, A, D\}$
        - The second step we delete second item D in the second cart, D exist in the first cart, so we delete it, $\{C, A\}$
        - The third step we delete third item E in the second cart, E doesn't exist in the first cart, skip it.
        - After these step, the result of difference should be $\{C, A\}$.

You should finish the following function, see the Function section below

Function

Constructor

You should complete these constructor below.

- ShoppingCart
  - `ShoppingCart()`
    - The default constructor of ShoppingCart class.
  - `ShoppingCart(int item_list_size, Product* item_list)`
    - The copy constructor can initilize the item list.
    - If the item list contains repeat item, you should throw a string exception.
- Product
  - `Product()`
    - The default constructor of Product class.
  - `Product(std::string name, int price)`
    - The copy constructor can initilize the product name and product price.
    - The length of name should longer than 4, the price should be a positive number, otherwise, you should throw a string exception.

Function

You should complete these function below.

- ShoppingCart
  - `int getItemCount() const`
    - Return the count of item in shopping cart.
  - `Product getItemByIndex(int index) const`
    - Return the product in shopping cart by index.
    - If the index is out of range, you should throw a string exception.
  - `void setItemByIndex(int index, Product product)`
    - Set the exist product in shopping cart by index.
    - If the index is out of range, you should throw a string exception.
    - If this action will appear duplicate item, you should throw a string exception.
  - `void appendItem(Product product)`
    - Append the product to the shopping list.
    - If the product already in the list, you should throw a string exception.
  - `int getTotalCost() const`
    - Return the total cost in shopping cart.
- Product
  - `std::string getName() const`
    - Return the name of product.
  - `int getPrice() const`
    - Return the price of product.

Operator

You should complete these operator below.

- ShoppingCart
  - `ShoppingCart& operator+(const ShoppingCart& other)`
    - Do the union by two shopping cart. (See Problem Content)
  - `ShoppingCart& operator-(const ShoppingCart& other)`
    - Do the difference by two shopping cart. (See Problem Content)
  - `ShoppingCart& operator=(const ShoppingCart& other)`
    - The copy assignment of ShoppingCart.
- Product
  - `Product& operator=(const Product& other)`
    - The copy assignment of Product class.
  - `bool operator==(const Product& other)`
    - The equals function of two Product class, to compare two product is equals or not.

Test

Midterm exam doesn't require you to write your own task, we already provide `ut_sample.h` to test your code.

The `ut_sample.h` depend the header `test_util.h`, which have a lot of utility to help complete the test case.

You can use the utility to write the test case if necessery.

Score

The score will calculate by the **successful percentage** of test suite,

e.g. Some one fail 5 test cases, we have 12 test cases in test suite, so the score will be $\lfloor(7/12)\times 15\%\rfloor = 8$.

- HW Part:
  - No score.
- TA Part:
  - Sample: 10%.
  - Product: 15%.
  - ShoppingCart_Regular: 15%.
  - ShoppingCart_Advanced: 60%.

You will see the score while the TA Job finished successfully, please check it.

Notice

- Use nullptr if you want to have a null pointer, which is a special pointer that doesn't point to anything.
- Use `ASSERT_EQ` to test integer, `ASSERT_NEAR` to test floating-point number, `ASSERT_THROW` to test exception, `ASSERT_TRUE` to test the conditional statement is true, or use `ASSERT_FALSE` to test when it should be false.
- Please implement your test cases reasonably, otherwise you will get no point for the task.
- You should neither add bin foler to your git, nor add a file with the name of '.gitignore' in bin folder (see our class repo).
- Some situation you will lose score:
  - **You lose 5 points for each test that has memory leak. You can check memory leak with `valgrind` cmd.**
  - **You will lost 10% if your bin folder contains compiled ut_all in git repo.**
- If you see segmentation fault, you can use `gdb` cmd to help debug. Link.

Meme

PEOPLE WHO GAVE YOU THE HINT TO PASS OOP MIDTERM
DON'T GONNA BUY TWO STUFF.
imgflip.com