OOP 2024S HW8 (due to 4/22 23:59)

TA 許澤群 u312510152.ee12@nycu.edu.tw

Problem

Design class Money, write the interface and implementation in sperate files (*money.h* and *money.cpp*). The private variable in Money is

$$allCents = dollars \times 100 + cents$$

What you need to design are (refer to appendix),

1. 4 operators for class Money: <, <=, >, >=

   The operators compare the *allCents* of Moneys.

2. Member function: *Money percent(int percentFigure) const*

   This function return a Money class whose *allCents* is *percentFigure* percent of Origin Money *allCents*. Discard decimal of *allCents* anyway. Example:

   m1=$100.10

   m2=m1. percent(10) = $10.01

TA will provide *main.cpp* and *money.h.* You need to add the definition of above operators and function into *money.h.* Then, create *money.cpp* by yourself and write down the statement in *money.cpp*.

Test case

Fisrt line of input file is the number or test case. For the follow test case, the first line is test *option*, which are 0: <, 1: <=, 2: >, 3: >= and 4: *percent* function. The second line are dollars and cents of Money1. And the third line are dollars and cents of Money2 or *percentFigure*.

```
6      // number of test case
0      // 1st case option (< operator)
100 10 // Money 1 dollars & cents
100 50 // Money 2 dollars & cents
1
50 30
30 50
2
30 50
50 30
3
70 20
70 20
4      // 5th case option (percent function)
100 10 // Money 1 dollars & cents
10     // percentFigure
4
70 50
25
```

```
1     // 100.1 <  100.5
0     // 50.3  <= 30.5
0     // 30.5  >  50.3
1     // 70.2  >= 70.2
10.01 // 100.1 * 10%
17.62 // 70.5  * 25%
```

Figure of example input(left) and golden(right)

Command

compile

```
g++ main.cpp money.cpp -I . -o Hw08
```

execute

```
./Hw08 input1.txt
```

OJ

```
/home/share/demo_OOP112_2 Hw 08
```

Submission

Compress *money.h* and *money.cpp* into <*Student ID*>_*HW8.zip* and submit it on newE3.

Appendix

3. Redo Practice Programs 1 from Chapter 11, but this time define the `Money` ADT class in separate files for the interface and implementation so that the implementation can be compiled separately from any application program.

1. Modify the definition of the class `Money` shown in Display 11.8 so that all of the following are added:

   a. The operators `<`, `<=`, `>`, and `>=` have each been overloaded to apply to the type `Money`. (*Hint:* See Self-Test Exercise 13.)

   b. The following member function has been added to the class definition. (We show the function declaration as it should appear in the class definition. The definition of the function itself will include the qualifier `Money::`.)

```
Money percent(int percentFigure) const;
//Returns a percentage of the money amount in the
//calling object. For example, if percentFigure is 10,
//then the value returned is 10% of the amount of
//money represented by the calling object.
```

   For example, if `purse` is an object of type `Money` whose value represents the amount $100.10, then the call

```
purse.percent(10);
```

   returns 10% of $100.10; that is, it returns a value of type `Money` that represents the amount $10.01.