METU NCC,
CNG 242, Spring 2010,
Homework 4

Due 8^{th} May 2010

In this homework you will implement an class similar to Homework 3 **Haskell** abstract data type Bill . Most of the homework description is similar to HW3 but implementation language is **C++**.

Bill datatype will keep track of a customers bill. At each bill there are 0 or more orders (consider a restaurant). Customer can order 1 or more pieces of any item. Each item has a unique unit price. Customer can order as many items s/he wishes. Items are given arbitrary string names. All strings operations are case sensitive.
You are required to implement the following class in bill.cpp:
`class Bill { ... }`
Bill class will have the following member functions implemented:

Bill();
It is a constructor initializing a new bill as an empty bill with no orders yet

Bill(const Bill &);
It is a constructor initializing a new bill from an existing bill object. It is copy constructor to enable Bill objects to be initialized from object, passed as a value and returned as a value. Copy semantics for heap variables are implemented.

Bill & add(string *name*, int *quantity*, double *unitprice*);
It adds an order to current bill. Current bill object is updated and reference to current object (`*this`) is returned. The first parameter (string) is the name of the item, second (int) is the number of pieces for the item, third (double) is the unit price of the item. If there exists an order with exact same name, the new order number of pieces is added to the current number of pieces and price is updated as the new price in third parameter. If no such order exists, it is inserted with is number of pieces and price values.

For example `Bill t1; t1.add("pizza",2,3.4)` adds 2 pieces of pizza with 3.4TL each on an empty bill and returns the bill.
`t1.add("pizza",3,3.2)` adds 3 more pieces of pizza to previous value. The resulting value has 5 pieces of pizza and price of each piece is 3.2 (later overrides).
`t1.add("pasta",1,3.2)` adds 1 piece of pasta in `t1`. The object `t1` has 1 piece of pasta (3.2TL) and 5 pieces of pizza ($5 \times 3.2 = 16$TL)

void get(string *name*, int & *quantity*, double & *uprice*)
It gets the current state of order for the given name. First parameter is the name of the item. It puts the number of pieces the name is ordered in pass by reference parameter *quantity* and the unit price of the order in pass by reference parameter *uprice*. If order does not exists in the bill object, 0 is put in *quantity*.

`t1.get("pizza",i,p)` puts 5 and 3.2 on `i` and `p` respectively. `t1.get("pide",i,p)` puts 0 on `i`.

Bill &quickadd(string);
It simply increments the number of pieces of the first parameter name. If name exists in the current bill, the exact bill with the number of pieces for the name is incremented. If it does not, it generates an error. In order to produce the error, if name does not exist in the bill, evaluate ‘`cerr << "not ordered yet\n"` . It will cause a error and execution continues. Function returns the reference to current object.

Bill &cancel(string *name*, int *quantity*);
It cancels some of the orders from the bill. The first parameter is the name of the item and the second parameter is the number of items to be cancelled. It returns the reference to current object.

If the name has larger number of pieces already ordered, the number of pieces are decremented by the second parameter. Otherwise, if order is less than equal to the parameter, all orders of given name are deleted. There is no error checking about if the parameter is larger than the existing number of pieces.

`t1.cancel("pizza",1);` decrements pizza count by 1 (it gets 4).
`t1.cancel("pizza",4);` returns a bill without any pizza (4-4 is 0).
`t1.cancel("pasta",200)` returns en empty bill, no order left.

Bill & operator += (const Bill & *otherbill*);

It merges two bills into current bill. It will contain both orders. The unit prices will be taken from the RHS parameter *otherbill* in case of conflict. It returns the current, updated bill reference.

`t1 += t2;` all orders in t2 are added in t1. t1 reference is returned.

double operator (double) ();

It calculates and returns the total price of the orders. It is sum of all item prices and each item price is the number of pieces multiplied by the unit price.

```
double x = (double) t1
```

Bill & operator= (conts Bill &) ();

Assignment operator implementing copy semantics. The heap variables inside the object are copied properly.

friend ostream & operator << (ostream & *ostr*, const Bill & bill);

Not a member function but declared as a friend to output all orders in the bill.
As a result, "`cout << t1 ;`" will cause orders in the bill to be displayed on standard output. The following is the format to use for your bills:

```
Name                    Pieces  U.Prc.   Price
-----------------------------------------------
ayran                        3    1.40    4.20
coffee                       2    2.00    4.00
orange juice                 1    4.00    4.00
pasta                        2    3.00    6.00
pizza                        2    3.40    6.80
puding                       1    3.10    3.10
-----------------------------------------------
Price:                                    23.81
VAT:                                       4.29
Total:                                    28.10
```

In order to get formatted string and numerical values:

```
#include<iomanip>
```

and use `cout << setw(6) << i << setw(8) << setprecision(2) << 123.23123131;`

Please note that the items are sorted based on names during output. You can either keep them sorted or sort during output. Show also displays the total sum, VAT (Value added tax, which is $\frac{sum*0.18}{1.18}$) and total sum without VAT. Orders are kept in menu prices, including VAT, then VAT calculated and shown during `show`.

You can use C++ Standart Template Library (which is standard in all C++ compilers) type `string` in the homework. In order to use, add `#include<string>`. `string` data type can be assigned, compared, concatanated using standart operators and you don't have to care about its size and allocation. **You are not allowed to use other STL classes**.

Sample code:

```
#include<iostream>
#include<hw4.h>

int main() {
        Bill a, b, c;
        int t,u;

        a.add("pizza",2,3.4);
        a.add("pasta",2,3.0).add("orange juice",1,4.0);
        a.add("pizza",1,3.2);

        cout << a << "\n--\n";

        b.add("ayran",2,0.9);
        b.add("coffee",2,1.5);
        b.add("pide",4,2.5).add("adana kebap",2,3.0).add("soup",2,1.4);
        b.add("orange juice", 1, 3.5);
        cout << b << "\n--\n";

        b.cancel("pide",2);
        b.cancel("coffee",3);
        cout << b << "\n--\n";

        a.get("pizza", t, u);
        cout << t << ':' << u << '\n';
        a.get("pide", t, u);
        cout << t << ':' << u << "\n--\n";

        c=a;
        c.quickadd("ayran");
        c.quickadd("orange juice");

        cout << a << "\n--\n";
        cout << c << "\n--\n";

        a += b;
        cout << a << "\n--\n";
        a += c;
        cout << a << "\n--\n";

        cout << (double) a << ':' << (double) b << ':' << (double) c << '\n';

        return 0;
}
```

Starting with this homework you will learn and use separate compilation. Put your class definition in `hw4.h` containing only member variables, **prototypes** of member functions, and friend declaration if required. Make only the required part specified as interface `public`. Everything else will be `private`.
In `hw3.cpp`, put your class definitions. In first line include `hw4.h` and other includes, than implement the member functions (that has prototypes in the header file) in this file (as `Bill::...`). Do not implement `main()` function.
On a different file like `hw4main.cpp` include `hw4.h` and implement `main()` function like the example above. In your development environment insert all in a new project, then compile.

Submission
Put your two files, hw4.h and hw4.cpp on a single **zip** file and submit the zip file. Post your submission on METU Online. You can ask questions on METU Online forums. Note that all homeworks should be result of you individual work. No code sharing is allowed. Cheating policy: cheaters (both parties) take 0 from all 5-6 homeworks. Commissioning/trying to commission someone else to do your homework is subject to disciplinary action.

Output

Output of the sample program above is:

```
Name                    Pieces  U.Prc.  Price
-------------------------------------------
orange juice                 1    4.00    4.00
pasta                        2    3.00    6.00
pizza                        3    3.20    9.60
-------------------------------------------
Price:                                   16.61
VAT:                                      2.99
Total:                                   19.60
--
Name                    Pieces  U.Prc.  Price
-------------------------------------------
ayran                        2    0.90    1.80
adana kebap                  2    3.00    6.00
coffee                       2    1.50    3.00
orange juice                 1    3.50    7.00
pide                         4    2.50   10.00
soup                         2    1.40    2.80
-------------------------------------------
Price:                                   25.93
VAT:                                      4.67
Total:                                   30.60
--
Name                    Pieces  U.Prc.  Price
-------------------------------------------
ayran                        2    0.90    1.80
adana kebap                  2    3.00    6.00
orange juice                 1    3.50    7.00
pide                         2    2.50    5.00
soup                         2    1.40    2.80
-------------------------------------------
Price:                                   19.15
VAT:                                      3.45
Total:                                   22.60
--
3:3.2
0:0
--
Name                    Pieces  U.Prc.  Price
-------------------------------------------
orange juice                 1    4.00    4.00
pasta                        2    3.00    6.00
pizza                        3    3.20    9.60
-------------------------------------------
Price:                                   16.61
VAT:                                      2.99
Total:                                   19.60
--
Name                    Pieces  U.Prc.  Price
-------------------------------------------
orange juice                 2    4.00    8.00
pasta                        2    3.00    6.00
pizza                        3    3.20    9.60
-------------------------------------------
Price:                                   20.00
VAT:                                      3.60
Total:                                   23.60
```

```
--
Name                    Pieces  U.Prc.  Price
-------------------------------------------
ayran                        2    0.90   1.80
adana kebap                  2    3.00   6.00
orange juice                 2    3.50   7.00
pasta                        2    3.00   6.00
pide                         2    2.50   5.00
pizza                        3    3.20   9.60
soup                         2    1.40   2.80
-------------------------------------------
Price:                                   32.37
VAT:                                      5.83
Total:                                   38.20
--
Name                    Pieces  U.Prc.  Price
-------------------------------------------
ayran                        2    0.90   1.80
adana kebap                  2    3.00   6.00
orange juice                 4    4.00  16.00
pasta                        4    3.00  12.00
pide                         2    2.50   5.00
pizza                        6    3.20  19.20
soup                         2    1.40   2.80
-------------------------------------------
Price:                                   53.22
VAT:                                      9.58
Total:                                   62.80
--
62.8:22.6:23.6
```