ITU COMPUTER ENGINEERING DEPARTMENT

BLG 252E OBJECT ORIENTED PROGRAMMING

HOMEWORK -3

Due Date: 8th of May, 2019 23:59 pm.

"If you really look closely, most overnight successes took a long time."

-- Steve Jobs

For this homework assignment, you will be designing a restaurant by using polymorphism. The restaurant has five empty tables, the tables could include more than one customer so more than one dishes/beverages can be ordered at the same time. There will be no limitations in the number of customers or the orders. The restaurant has three types of ingredients (type1, type2 and type2) to be served in different types of quantities (grams, units or milliliters).

You are also expected to write a checkout system as well as different classes to simulate this restaurant. The checkout system will simply calculate the bill to be paid (total cost + tax + tips) and to be printed out on the screen.

There are three types of ingredients to be served:

- **Type1** has a *weight* and a *price per grams* which are used to determine its *cost*. (e.g. 200 grams of chicken)
- **Type2** has a *number* and a *price per unit* which are used to determine its *cost.* (e.g. 2 onion)
- **Type3** has a *milliliter* and a *price per milliliter* which are used to determine its *cost.* (e.g. 50 ml olive oil)

You will also be given three .txt files (stock.txt, menu.txt and order.txt) in advance. Stock.txt and menu.txt will remain exactly the same but a different order.txt file will be used to grade your homeworks with ***CALICO***.

stock.txt: This file should include the restaurant's stock information. The name of item, the type of item, item count in types of *item type* and price of the item per unit (of *item type).* (all the items are separated by tabs)

An example of stock.txt is shown:

Name	Type	ItemCount	Price
chicken	1	750	0.05
beef	1	750	0.44
onion	2	10	2
fanta	2	4	3
chili	3	150	0.04
mustard	3	200	0.03
.	.	.	.
.	.	.	.

For example in the above sample stock.txt; there are 750 grams of chicken, 10 units of onion and 150 ml chili available in restaurant's stock. Also for example, if you need 250 grams of chicken to cook some dish the cost will be 250*0.05 = 12.5 TL. Or if you need 2 onions the cost will be 2*2 = 4 TL.

menu.txt: This file should include the restaurant's menu information. The name of dish, the ingredients if available (the salads and beverages do not have ingredients as they are all ready to be served and sold in fixed price given in stock.txt). (all the items are separated by tabs, but ingredients are separated by comma)

An example of menu.txt is shown:

Name	**Ingredients**
massala chicken	250 gram chicken, 3 onion, 2 tomato,1 lemon, 1 garlic, 10 ml olive oil
mughai chicken	400 gram chicken, 4 onion, 6 garlic, 1 lemon, 10 cashew, 10 ml chili
……	…….
ceaser salad	N/A
bbq salad	N/A
……	…….
coke	N/A
fanta	N/A
……	…….

For example in the above sample menu.txt; the customer can choose any number of dish, salads or beverages if available. There are no limitations in the number of orders, the main dishes should be checked from stock if they could be prepared with the given ingredients lists (all ingredients must be met before the dish could be prepared) and served. After serving the dish, you must update the stock information so that for the next order it could be checked if there are still available supplies. (Meaning if the table 1 ordered masala chicken, for the table 2's order the ingredients must be updated as you already used some items from stock) If there are lack of ingredients then an error message must be thrown (e.g *"We don't have enough bbq salad" or "We don't have enough massala chicken"*). For the salads and beverages the program should only check the existence of these items in the stock.

order.txt: This file should include the restaurant's order information from the customer side. It will include name of table, number of lines to be read, and list of items ordered.

An example of order.txt is shown:

```
Table1
2
2 massala chicken
1 coke
Table2
3
1 mughai chicken
1 fanta
1 tea
Table3
1
3 bbq salad
```

For example in the above sample order.txt;

- from table1: you will read two lines (2 massala chicken and 1 coke),
- from table2: you will read three lines (1 mughai chicken, 1 fanta, 1 tea),
- from table3: you will read one line (3 bbq salad).

By reading these three .txt files you will design and implement a restaurant program to meet the customers need and calculate the checkout system. The program should first get customers' orders from **order.txt** and check the recipe from **menu.txt** and check from **stock.txt** their existence and if all the orders are available you can cook, and calculate the remaining ingredients and update the stock after that you can calculate the cost.

Let's assume;

Price =P,

total cost calculated while cooking the dish = C,

tax rate (static defined 8%) = tX,

Tip (static defined 15 %) = tp,

The checkout will be calculated as with the:

$$P = C + (C * \frac{tX}{100}) + (C * \frac{tp}{100})$$

The checkout system to be printed out to screen (let's assume only 3 tables orders are in the orders.txt) :

```
Table1 ordered:
2 massala chicken cost: 43
1 coke cost = 3
Tip is 6.9
Total cost: 53.34 TL
**************************
Table2 ordered:
We don't have enough mughai chicken
1 fanta cost = 3
1 tea cost = 2
Tip is 0.75
Total cost: 6.15 TL
**************************
Table3 ordered:
3 bbq salad cost: 30
Tip is 4.5
Total cost: 36.9 TL
**************************
```

Important:

- FIFO principle will be adopted for the order.txt. Meaning the program should try to meet the table of orders in the way you read from the order.txt file. If any of items is sold out it should print an error message.
- Item types will be designed by polymorphism structure.
- All the dynamic data members should be declared as private.
- You should successfully deallocate all of the allocated memory before termination of your program.
- Do not forget to handle exceptions and print error messages.
- Use comments whenever necessary to explain your code. Also, write your name on top of each file you are sending in comment!
- Do not forget all the checkouts must be written on screen.

SUBMISSION

1. Make sure you write your name and number in all of the files of your project, in the following format:

```
/* @Author

 * Student Name: <student_name>
 * Student ID : <student_id>
 * Date: <date> */
```

2. Use comments wherever necessary in your code to explain what you did.
3. **Your program should compile and run on Linux environment using g++ (version 4.8.5 or later). You can test your program on ITU's Linux Server using SSH protocol.**
   To compile the code, you can use the following command:
   g++ -c -std=c++11 main.cpp -o main.o
   or
   g++ -c main.cpp -o main.o
   Both compilations will be acceptable but please make sure you write your runnable code in main.cpp, If it cannot be compiled and linked using this command, it will not be graded (failed submission).
   Your program will be checked using ***CALICO*** (https://bitbucket.org/uyar/calico) automatic checker. Therefore, make sure you **print the messages exactly as given in the homework definition.**
   An example calico.yaml file is already given for you to test your results.
4. After you make sure that everything is compiled smoothly, archive all files into a zip file. Submit this file through www.ninova.itu.edu.tr. Ninova enables you to change your

submission before the submission deadline.

Do not miss submission deadline. **Do not** leave your submission until the last minute. The submission system tends to become less responsive due to high network traffic.

HOMEWORKS SENT VIA E-MAIL WILL NOT BE GRADED.

Academic dishonesty including but not limited to cheating, plagiarism and collaboration is unacceptable and subject to disciplinary actions. Your homeworks will be checked with a plagiarism checker system, any student found guilty will receive 0 as his/her grade for the homework and subject to disciplinary actions.

If you have any question about the homework or you are confused, DO NOT hesitate to contact the teaching assistant **Dilara TORUNOĞLU SELAMET** via e-mail (**torunoglud@itu.edu.tr**) or in **4307**. In fact I encourage you to ask questions ☺

Good Luck!