Software Engineering Design – Advanced Programming Techniques, Semester 2022A

RMIT UNIVERSITY

LAB ASSESSMENT 2 (35%) – MOCK TEST QUESTIONS

Test Duration: 150 mins (+ 15 mins for submission)

NOTE: only do and submit **one source .cpp file for each question 1-3 (three files for three questions), and don't zip them together.**

Read through all the questions and do the easiest ones first.

Question 1 (11 pts)

Define a class namely **Acc** with an attribute is ***name*** (*string*), ***bill*** *(total amount need to pay: double),* and a method namely ***buyProduct****(int price)* which will add up the product's price to the ***bill***.

Define another class **GoldAcc** which inherits from the class Acc, with an extra attribute named **discRate** (*discount rate: double). Override the* ***buyProduct*** *method for* GoldAcc so that the price will be discounted by discRate before adding to the bill

Example: price = 1000, discRate = 15% → only add 1000*85% = 850 to the bill.

Provide suitable constructors and test **buyProduct**() methods of both Acc and GoldAcc classes in main().

Question 2 (12 pts)

Use linked list concepts to record real estate transactions (selling and buying a house) as below.

- David: initially bought the house for $800
- David --> John : price = $1000
- John --> Peter : price = $1200
- Peter --> Luna : price = $1800
- Luna --> Sophia: price = $3500

Hint: Define a class, e.g. namely **Broker**, with attributes are ***name***, ***buyPrice***, ***sellPrice*** and ***nextBuyer***.

a) Write a function to print out all transactions exactly as above

```
David: initially bought the house for $800
David --> John :  price = $1000
John --> Peter :  price = $1200
Peter --> Luna :  price = $1800
Luna --> Sophia:  price = $3500
```

b) Write a function to print out information of the brokers with the **lowest** and **highest profit** (note: *profit = sellPrice - buyPrice*).

c) Write *a function to allow deleting a transaction* within the linked list. Test it in main().

Question 3 (12 pts)

Reuse the code from Question 1 and further upgrade it for the requirement below:

The investor of the app requires that it must manage each **product** with *name* and *price*. The app also need to manage the **shop** who selling the products (shop's name and list of selling products).

For each customer account, besides total bill value, we also need to manage *list of bought products*. In addition, the customer can return the product (which will be charged with a fee 10% for the product price for normal accounts, but only 5% for gold accounts) to the shop that sell that product.

Implement classes with suitable attributes and methods to satisfy the above requirement. Test them in main() with appropriate output messages.