Software Engineering Design
Advanced Programming Techniques
Semester 2, 2024

RMIT UNIVERSITY

LAB ASSESSMENT 2 (30%) – TEST QUESTIONS

Test Duration: 120 mins (+ 15 mins for submission)

NOTE: only submit **one .cpp file for each question** (three files for three questions), and **DON'T zip** them together.

Question 1 (6 pts)

A blockchain is a data structure linked by a hash pointer to the previous block. The pointers point to the previous hash, as shown in Figure 3.1.

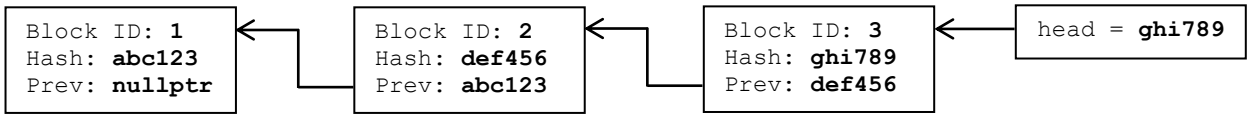

Figure 3.1: Blockchain diagram

The code to build our chain from the data file **chain.dat** produces an error (refer to the assessment question link to download code and data files).

(a) Debug and fix the error.
(b) Complete the method **showChain** to output the linked list to the console.

Question 2 (12 pts)

Write a C++ program to manage reservations and billing for a buffet restaurant that offers various meal options at different prices. For example, the restaurant may provide meals such as breakfast, lunch and tea, priced at $12, $20 and $10 per person, respectively. The program should include methods to make reservations, display details of a specific transaction, and show overall statistics. Details are as follows:

The restaurant stores its ***name*** and maintains ***a list of transactions***. Each ***transaction*** records the details of a reservation, including the following: ***Transaction ID*** (which is unique and counting up from 1 - refer to the example indicated by the red circle in the sample run) and the ***reserved meal detail***. The ***reserved meal detail*** contains the ***meal type*** (for example, breakfast, lunch or tea), ***number of people*** for the reservation, and the ***total price calculated*** of the reservation. The ***meal type*** has the ***type*** and the ***price per person*** for each type.

The restaurant provides the following three methods:

(a) **makeReservation** to calculate the total price of the reservation based on the meal type and the number of people. It then adds the reservation to the transaction list with a unique transaction ID.

If the number of people exceeds 10, a 10% discount is applied. You may assume that the inputs entered are correct.

(b) **showTransaction** to display the details of a specific transaction by its transaction ID.

(c) **showStats** to display the total number of reservations and the total revenue collected for each meal type.

Implement classes with suitable attributes and methods to satisfy the above requirements. Test them in main() with appropriate output messages. Your sample output should be the same as the sample run shown below. ***Note****: all attributes should be declared as* ***private****. You can utilize friend class if needed.*

Sample run:

```
Welcome to RMIT Restaurant

Meal type and Price per person
==============================
Breakfast, $12.00
Lunch, $20.00
Tea, $10.00
==============================
Transaction ID 1 has been successfully added. Total Price: $40.00
Transaction ID 2 has been successfully added. Total Price: $108.00
Transaction ID 3 has been successfully added. Total Price: $60.00

Transaction ID: 1
Meal Type: Lunch, Number of People: 2, Total Price: 40.00

Transaction ID: 2
Meal Type: Tea, Number of People: 12, Total Price: 108.00

Transaction ID: 3
Meal Type: Lunch, Number of People: 3, Total Price: 60.00

Restaurant Statistics
Breakfast Reservations: 0, Total Revenue: $0.00
Lunch Reservations: 5, Total Revenue: $100.00
Tea Reservations: 12, Total Revenue: $108.00
```

Question 3 (12 pts)

Figure 3.2 shows a **singly linked list** containing non-repeating positive integers and two occurrences of negative 1 (i.e., -1).

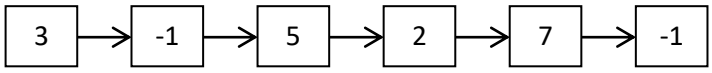

Figure 3.2: Singly linked list

(a) Define a class called ***Analyser***, with private attributes ***value*** and ****nextNode*** that connects to the next node. Create the singly linked list as shown in Figure 3.2.

(b) Write a method **traverseList** to traverse the linked list and print only the node value(s) in between the two occurrences of -1. Refer to the example output of Figure 3.2 in the sample run.

(c) Write a method **removeNeg** that removes both occurrences of -1 from the linked list, and then traverses and prints the values of all the nodes in the updated list.

(d) Write a method **printMin** that prints the minimum value among all the nodes in the linked list.
Note: printMin should be called after **removeNeg** has been completed.

Test all the methods in **main()** with appropriate output messages.

Sample run:

```
Linked list after traverseList is called
5-->2-->7-->

Linked list after removeNeg is called
3-->5-->2-->7-->

Min is 2
```

--- End of Paper ---