**COMP6048001 - Data Structures**

**QUIZ 1 – 15 February 2023**

1. **[20 points]** Given the following mathematical statement in the form of Infix, using Stack Method, convert those statements into the Prefix. Please use the given table as reference for each operator precedence. Higher precedence level means more priority over the other.
2. **[10 points]** A ^ B / (C – D)
3. **[10 points]** ( A - ( B \* ( C + D ) ) )

**Operator Precedence Table:**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Notes** | **Precedence** |
| + | Addition | 1 |
| - | Subtraction | 1 |
| \* | Multiplication | 2 |
| / | Division | 2 |
| ^ | Exponentiation | 3 |

1. **[30 points]** Given the following Binary Search Tree:
   1. **[15 points]** Do insert process for key values **30** and **28** respectively. Show the final tree!
   2. **[15 points]** From the result of 2a, do delete process for key value **21** and **47** respectively. Show the final tree!
2. **[50 points] Hov’s Fresh Concoction Study Case.**

You are given a chance as a junior internship in one of the growing beverage companies: **Hov’s Fresh Concoction**. They asked you to create a **point of sales application** for their offline store using **C Programming Language**. They also asked you to implement **Linked List** in your algorithm. The requirement are as follows:

1. **Application Overview**

* You have access to two files: **“customers.txt”** and **“menus.txt”**
* Inside the **“customers.txt”** file you can find all **customer** data which already have **membership**. Each row in the file contains: **Customer ID, Customer Name,** and **Customer Type of Membership**.
* There are **five types of membership**, in **ascending** **order**: **"Non-Member", "Bronze", "Silver", "Gold", "Platinum"**. **Higher** **membership** will place **their order sooner** in the **queue**.
* Inside **“menus.txt”** you can find all **menus** available in the store. Each row in the file contains: **Menu ID, Menu Name, and Menu Price**.
* **Without these two files, the application cannot run properly**.
* There are **four** **main** **menus** in the application. Each will be explained in the next section.
* **For your convenience, please check the given exe file as reference**.

1. **Menu 1: Add New Order**

* This menu will be used to **add order to the queue**.
* First, the user needs to input the customer **name** then the application will check whether the given name is a member or not. If member data is found, then it shows details of their membership.
* After that, show all available menus. Users can **add items** to their order by entering **Menu ID** and the **quantity**. To end the order process, users need to input -1. **Make sure to give proper validation for every input process**.
* After finishing entering the order, then **show a summary of their order**. The summary must include a list **of items in their order**, **subtotal**, and **total**.
* System then will **generate** an **Order** **ID** with the following format:

**0[Membership Type ID]-[A-Z][A-Z][0-9][0-9][0-9][0-9]**

[A-Z] is random character between A to Z

[0-9] is a random number between 0 and 9

For example, a gold customer might get 04-BQ5959 as their Order ID.

* After that **entry** **the order into the queue system** by **considering** **their** **priority**, based on **customer** **membership** **type**. **For the same membership type, earlier orders will be processed first**.

1. **Menu 2: View Order Queue**

* Users can use this menu to **show** **all** **orders** in the system.
* For each order, show the **customer** **data** and **detail** of **their** **order**.

1. **Menu 3: Serve Order**

* Using this menu, will **indicate** that **the first order in the queue is finished**.
* Show the **Order** **ID** and the **Customer** **Name**.
* Then **remove** the **data** **from** **the** **queue**.

1. **Menu 4: Exit**

* Using this menu will first **remove all data from the queue**.
* After that the **application** will be **terminated**.

**--Good Luck and Do Your Best!!!--**