


<b>Practicum Case</b>	
COMP6048   COMP6048001   COMP6048016   COMP6048049 Data Structures	
<b>Computer Science</b>	<b>O221-COMP6048-AM01-09</b>
<i>Valid on Even Semester Year 2021/2022</i>	<b>Revision 00</b>

### Learning Outcomes

- LO1 – Explain the concept of data structures and its usage in computer science
- LO2 – Illustrate any learned data structures and its usage in application
- LO3 – Apply data structures using C

### Topic

- Session 9 – AVL Tree

### Sub Topics

- Push
- Update
- Search
- Pop
- Pop All

**Soal**  
Case

## Bluejack GShop

**Bluejak GShop** is one of the most popular offline game stores in town. Recently they want to open a new branch in another city. They need a programmer to help them create a program that can help them manage game stock in their warehouse. The criteria of the program are:

- The program consists of **4 menus**, there are:

1. **Insert Game**
2. **View Game**
3. **Update Stock**
4. **Exit**

```
Bluejack GShop
=====
1. Insert Game
2. View Game
3. Update Stock
4. Exit
>> █
```

Figure 1. Main Menu

- If the user chooses **View Game (Menu 1)**, then:
  - Ask user to input the following data:
    - **Game Title**
      - Validate the inputted game title must be **between 5 and 25 characters**.
      - Validate the inputted game title must be **unique**.
    - **Game Genre**
      - Validate the inputted game genre must be **either** “Action”, “RPG”, “Adventure”, or “Card Game”.
    - **Game Stock**
      - Validate the inputted game stock must be **at least 1**.
  - Then, push the data to the **AVL Tree** with **game title** as **key**.

```
Input game title[5-25][unique]: Dota
Input game title[5-25][unique]: Dota 4
Input game type[Action|RPG|Adventure|Card Game]: adventure
Input game type[Action|RPG|Adventure|Card Game]: Action
Input game stock[>= 1]: 0
Input game stock[>= 1]: 50
Insert success !
```

Figure 2. Insert Menu

- If the user chooses **Insert Game (Menu 2)**, then:
  - Validate if the data is **empty**, then show “**Warehouse is empty !**” message.

```
Warehouse is empty !
Press enter to continue...
```

Figure 3. Data Empty Message

- Otherwise, show all the game using **In-Order method**.

```
-----
| Game Title          | Game Genre  | Game Stock |
-----
| Dota 4              | Action      | 50         |
-----
| Light of The Tomb Rider | Adventure   | 27         |
-----
| Warcraft 4          | RPG         | 75         |
-----
Press enter to continue...
```

Figure 4. Show Game Data Using In-Order Method

- If user chooses **Update Stock (Menu 3)**, then:
  - Ask user to input **game title**. Validate the inputted game title must **exist** in the **AVL Tree**.  
**Otherwise, show “Data not found” message and redirect back** to main menu.

```
Input game title: Dota 2
Data not found !
Press enter to continue...
```

Figure 5. Data Not Found Message

- Then, ask user to input **update type**. Validate the inputted type must be **either “add” or “remove” (case insensitive)**.
- After that, ask user to input the **quantity** to add or remove. If user **chooses “remove”**, validate the inputted quantity must be **between 1 and current stock**. If user **chooses “add”** validate the inputted quantity must be **at least 1**.
- **Remove or add** the current stock with the inputted **quantity**. If user **remove all the remaining stock**, then **delete the data** from **AVL Tree**.

```
Input game title: Dota 4
Current stock: 50

Input update type[add|remove][case insensitive]: remove
Input stock to remove[1-50]: 51
Input stock to remove[1-50]: 50
Data updated successfully !
Dota 4 is removed from the warehouse !

Press enter to continue...
```

Figure 6. Data Removed Due To Out Of Stock

- If user chooses **Exit (Menu 4)**, **terminate** the program.

**Please run the EXE file to see the sample program**