Mr. X, a man with a gaming hobby, has many games. Because he has too many games to track, he decided to store the game names that he owns. To make searching efficient, he decides to use BST as the storing system.

He decides that that BST has some properties:

* Each node in BST stores **game name**, **game rating (1 – 10), played hours.**
* BST will sort the data based on **game name** (string) **alphabetically in DESCENDING ORDER** and **case insensitive**.

Here are the operations that can be applied:

1. Display (Not a menu)

In the main menu, before the menu list is displayed, **all game name, rating, played hours** are displayed using **Inorder style**.

1. Insert

If X chooses this menu, the program will request **game name**, **game rating, and played hours**. If **there is an** **existing game** **name** in BST, **update** the rating instead.

* Validate the game name has **minimum length of 1 and maximum length of 100 & alphanumeric**.
* Validate the game rating value has **range between 1 and 10 & numeric.**
* Validate the played hours value has **range between 0 and 2000 & numeric**.

1. Delete

If X chooses this menu, the program will inquire about the **game name** to be deleted.

* Validate the game name has **minimum length of 1 and ma`ximum length of 100 & alphanumeric**.
* If the game name is **NOT found**, display the message “Game name does not exist!”.
* If the game name is **found**, delete that node using the BST deletion method.

1. View by Rating

If X chooses this menu, the program will display all **game name**, **game rating, and played hours** sorted by **rating**. FOR games that has **the same rating**, the program will display by **FIRST IN FIRST OUT (FIFO)** rule.

Note: **DO NOT create extravagant design for the program**, **just simply display what is required is enough!**

The following image depicts the sample BST Mr. X has in mind:

