AVL Battle Royale

Description

You are developing a new battle royale game called "AVL Battle Royale". Inside the game, there are some players, each having their own *health points* (HP) which is a positive integer.

During a game, there will be Q scenarios. In each scenario, you will be given the following information:

- Scenario Type, which can either be a plus or minus.
- Player's name, represented by a string (at most 10 characters) consisting of only lower-case characters. Denote this variable as p.
- Quantity, which is a positive integer. Denote this variable as x.

In a plus-type scenario, the following will happen.

- If p is in the game, then increase their HP by x. Suppose that h represents their HP after the increase. Report by output "Heal player p, current health: h".
- Otherwise, insert a new player to the game with the quantity as their initial HP. Report by output "Player p is entered to the game".

In a minus-type scenario, the following will happen.

- If p is in the game, then decrease their HP by x. Suppose that h represents their HP after the decrease. There are two possible outcomes:
    - If $h > 0$, then output "Damage player p, current health: h".
    - If $h \leq 0$, then output "Player p is removed from the game".
- Otherwise, output "Player p is not in the game".

Task

You are tasked to implement the game. Use an AVL Tree with the player's name as the key to maintain all players that are currently in the game with their corresponding HP. Therefore, if a player is entered to the game, you need to insert them to the AVL Tree. Similarly, if a player is removed from the game, you need to delete them from the AVL Tree.

Your answer will be graded as follows: **30 points** for the AVL Tree Insertion, **30 points** for the AVL Tree Deletion, and **40 points** for the rest of your codes (including the main function and any additional AVL Tree Features that you need to implement the game).

Input/Output Format

First, you will be given an integer Q, which represents the number of scenarios.

For each scenario, you will be given a character, a string, and a positive integer in a single line. The character can either be + (representing the plus-type) or - (representing the minus-type). The string will consist of at most 10 characters, each of them is a lower-case character.

After the input for each scenario, your code should perform everything as described in the Description section and output the report accordingly.

Sample Input/Output

Sample Input	Sample Output
13 + aura 100 + jimmy 30 + duck 20 - aura 40 - quik 10 + aura 100 - duck 20 - duck 100 + duck 10 - jimmy 5 + quik 100 + duck 20 - duck 200	Player aura is entered to the game Player jimmy is entered to the game Player duck is entered to the game Damage player aura, current health: 60 Player quik is not in the game Heal player aura, current health: 160 Player duck is removed from the game Player duck is not in the game Player duck is entered to the game Damage player jimmy, current health: 25 Player quik is entered to the game Heal player duck, current health: 30 Player duck is removed from the game