Design a Leaderboard class, which has 3 functions:

1. addScore(playerId, score): Update the leaderboard by adding score to the given player's score. If there is no player with such id in the leaderboard, add him to the leaderboard with the given score.
2. top(K): Return the score sum of the top K players.
3. reset(playerId): Reset the score of the player with the given id to 0 (in other words erase it from the leaderboard). It is guaranteed that the player was added to the leaderboard before calling this function.

Initially, the leaderboard is empty.

**Example 1:**

**Input:**

["Leaderboard","addScore","addScore","addScore","addScore","addScore","top","reset","reset","addScore","top"]

[[],[1,73],[2,56],[3,39],[4,51],[5,4],[1],[1],[2],[2,51],[3]]

**Output:**

[null,null,null,null,null,null,73,null,null,null,141]

**Explanation:**

Leaderboard leaderboard = new Leaderboard ();

leaderboard.addScore(1,73); // leaderboard = [[1,73]];

leaderboard.addScore(2,56); // leaderboard = [[1,73],[2,56]];

leaderboard.addScore(3,39); // leaderboard = [[1,73],[2,56],[3,39]];

leaderboard.addScore(4,51); // leaderboard = [[1,73],[2,56],[3,39],[4,51]];

leaderboard.addScore(5,4); // leaderboard = [[1,73],[2,56],[3,39],[4,51],[5,4]];

leaderboard.top(1); // returns 73;

leaderboard.reset(1); // leaderboard = [[2,56],[3,39],[4,51],[5,4]];

leaderboard.reset(2); // leaderboard = [[3,39],[4,51],[5,4]];

leaderboard.addScore(2,51); // leaderboard = [[2,51],[3,39],[4,51],[5,4]];

leaderboard.top(3); // returns 141 = 51 + 51 + 39;

**Constraints:**

* 1 <= playerId, K <= 10000
* It's guaranteed that K is less than or equal to the current number of players.
* 1 <= score <= 100
* There will be at most 1000 function calls.