Amplitude Hackathon '23 Problem 3 - Dedupe

Time limit: 7.0s **Memory limit:** 1G

Amplitude customers are requesting a feature to deduplicate duplicate events that they accidentally send to Amplitude. It's your job to implement this service!

More specifically, you will receive events one by one - each event will have an alphanumeric ID and an expiration timestamp. Two events are considered to be the same if their IDs are the same. If an event is received and it has never been seen before, or the timestamp of its arrival exceeds its expiration timestamp when it was last accepted, then the event is accepted and its expiration timestamp is updated. Otherwise, the event is rejected.

Determine for each event whether it should be accepted or rejected.

Constraints

$1 \leq N \leq 10^6$

All IDs have lengths at most 10.

$1 \leq t_a, t_e \leq 10^{18}$.

If event i comes before event j in the input, then event i's t_a value is less than or equal to event j's t_a value. Additionally, if they have the same ID, then event i's t_a value is strictly less than event j's t_a value.

The expiration timestamp will always be greater than or equal to the arrival timestamp.

Subtask 1 [1 point]

$1 \leq t_e \leq 10^6$

Subtask 2 [1 point]

No additional constraints.

Input Specification

The first line of input contains a single positive integer, N.

Each of the next N lines contains an alphanumeric string of length at most 10 followed by t_a and t_e. The alphanumeric string is the ID, t_a is the arrival time, and t_e is the expiration time.

Output Specification

Output N lines. On the ith line, output `ACCEPTED` if the ith event is to be accepted. Otherwise, output `REJECTED`.

Sample Input 1

```
5
event1 1 2
event1 2 3
event1 3 4
Event1 4 5
Event1 6 7
```

Sample Output 1

```
ACCEPTED
REJECTED
ACCEPTED
ACCEPTED
ACCEPTED
```

Sample Explanation

The first event received is `event1`. The second event received is the same as the first one, and the arrival time (2) is less than or equal to the expiration time (2) so it is rejected. The third event received is also the same as the first one, but its arrival time (3) is greater than the expiration time. Note that the expiration time of the second event is ignored because the event was rejected.

The fourth event is `Event1`, which is distinct from the first event as IDs are case-sensitive. The fifth event is the same as the fourth event but is also accepted.

Sample Input 2

```
2
bigtime 9999999999 10000000000
bigtime 10000000001 10000000002
```

Sample Output 2

```
ACCEPTED
ACCEPTED
```

Sample Explanation 2

The timestamps can get very big here. Java users may need to use 64-bit integers to store the timestamps.