HASHING DATA STRUCTURE

- **Winner of an election**

- **array of names (consisting of lowercase characters)**

  Print the name of candidate that received Max votes

- **If there is tie, print lexicographically smaller name.**

```
Input:
n = 13
Votes[] = {john,johnny,jackie,johnny,john
jackie,jamie,jamie,john,johnny,jamie,
johnny,john}
Output: john 4
Explanation: john has 4 votes casted for
him, but so does johny. john is
lexicographically smaller, so we print
john and the votes he received.
```

Hello world

HASHING DATA STRUCTURE

- Now, we traverse the
- We need a unordered_map Name -> votes
- Create an Unordered_map

```
Input:
n = 13
Votes[] = {john,johnny,jackie,johnny,john
jackie,jamie,jamie,john,johnny,jamie,
johnny,john}
Output: john 4
Explanation: john has 4 votes casted for
him, but so does johny. john is
lexicographically smaller, so we print
john and the votes he received.
```

john -> 4

johnny -> 4

jamie -> 3

Jackie -> 2

Hello world

HASHING DATA STRUCTURE

- **Now, we traverse the unordered_map**

  Initialise two variable name and max_vote

- **1) Now take the key and its value**

  2) compare if (value > max_vote)

  3) store the max_vote = value;
  Name = key

  4) compare if (value == max_vote)

  check key < name
  Name = key

Hello world