What's your preference?

This étude is about a group of voters choosing a single candidate (e.g., a president) in an election. There are many voting systems that allow such choices to be made – you can read about some of them at the ACE Electoral Knowledge Network or through the Electoral Council of Australia and New Zealand. We'll discuss some of the computing-adjacent issues in elections in a town hall but this étude is about implementing the skeleton of a system for counting in the single transferable vote (STV) framework which is widely used in Australia and increasingly so in New Zealand as well.

Note that the STV can be, and frequently is, used in elections where multiple candidates are being selected, but the method of transferring votes in that case can be quite complex – and issues with infinite recursion can arise! For more details, the Wikipedia page on counting single transferable votes is quite interesting (even if it does have 'issues').

Problem Statement

The assumptions of the STV scenario that we will be modelling are as follows:

- The election is to choose one candidate from a pool of candidates.
- Each voter submits a list of (some or all of) the candidates in their order of preference.
- Votes are counted in a series of rounds until a winner is determined or an unbreakable tie occurs (see below).

In each round there is a pool of active candidates. Each voter's list of preferences is considered and their vote (if any) is assigned to the first candidate on their list who is active. If any candidate receives strictly more than 50% of the votes assigned then they win immediately. Otherwise, the candidate with the fewest votes is removed from the active pool and another round takes place.

Tie-breaking

What happens in a round if there are two or more candidates with the fewest votes?

We will use backwards tie-breaking. That means, we work backwards through previous rounds until we find some way to distinguish among the group of tied candidates. For instance if we had three candidates, A, B and C in this round who all received the least number of votes, we'd look backwards one round - if A got 12 there and B and C got only 8 we'd definitely keep A active, but then look backwards further to find which of B and C we should eliminate.

For the purposes of this exercise, if we fail to find a round in which B and C can be distinguished we would declare the tie unbreakable (in practice, either both would be eliminated or one would be chosen randomly).

We tie-break in the same way if there are exactly two active candidates and they receive the same number of votes.

Task

This is a 'proof of concept' task, so some of the underlying assumptions are quite unrealistic for a real election. As usual the program will take input from `stdin` and write output to `stdout`

Input

Input will be a sequence of lines containing names. A name is any sequence of non-whitespace characters. Within a line, names are separated from one another by whitespace (could be a single place, a tab, or multiple spaces, or ...)

Each line represents a single voter's preference (there may be empty lines representing spoiled ballots or something like that).

The candidate pool in the election consists of all the names that occur at least once in any voter's list.

Output

The output is to be a report on the process of the election. Each round of counting produces a block of output. The first line of this block is `Round <n>` where `<n>` stands for the number of the round (starting from 1). This is followed by one line per active candidate giving the candidate's name and their total votes this round. These should be ordered in decreasing order of number of votes, and alphabetically by candidate for candidates receiving the same number of votes in a round. As shown in the example below, the numbers should be left justified in a single column beginning three spaces past the end of the longest name that occurs. The last line of the block is one of the following: `Eliminated: <name>`, `Winner: <name>` (with an appropriate substitution for `<name>`) or `Unbreakable tie`, . Rounds are separated from one another by blank lines and the output ends when a winner is declared or an unbreakable tie occurs.

Example

Input:

```
Uohwjfk Bhnnfiu Edodq Zouqitwbc
Bhnnfiu Uohwjfk Zouqitwbc Edodq
Edodq Zouqitwbc Uohwjfk
Uohwjfk
Uohwjfk Bhnnfiu
Zouqitwbc Edodq Bhnnfiu
Zouqitwbc
Zouqitwbc Uohwjfk Bhnnfiu
Uohwjfk Bhnnfiu Edodq
Edodq
```

Output:

```
Round 1
Uohwjfk     4
Zouqitwbc   3
Edodq       2
Bhnnfiu     1
Eliminated: Bhnnfiu

Round 2
Uohwjfk     5
Zouqitwbc   3
Edodq       2
Eliminated: Edodq

Round 3
Uohwjfk     5
Zouqitwbc   4
Winner: Uohwjfk
```

Limits

Your program should not make any assumptions about maximum or minimum numbers of voters or candidates. In practice, I won't be testing with more than a million voters or a thousand candidates.

(points, Individual)