**New dating app**

**Note: this question will be graded for 1%.**

Little Bug believes in pure predestination, not math or luxurious profile to find her life partner. She wants to build a dating system and asks for your help.

The app she builds is called Spotifind (because she likes Spotify), it will match men and women for a blind date as she believes people can fall in love solely in each other’s souls. Her design works like this:

* Users of this app are in one of two genders: male and female.
* At any point in time, users can submit a **date request** to find a person of the opposite gender to date.
* Little Bug thinks that earlier man matches earlier woman, so the app actually matches users in **first come first serve** manner. Whenever a user submits a **date request**
  + If there is no person of the opposite gender waiting to be matched, this user will wait for his/her other half to come.
  + If there are some users of the opposite gender waiting, the app will immediately match this user with the waiting person who submits the *earliest* **date request.**

Your task is to implement the logic of this app. You will receive **date requests** from users via standard input, and whenever a match is made, you should output them to the standard output.

Also, Little Bug wants to evaluate the average *waiting time* of **matched users. Matched users** are those who luckily get a match**.** The *waiting time* of a user who is matched is the **number of requests** made into the app after this user submits a **date request** until this user is matched (including the request that matches this user). Please see the example below for further clarification.

Utilization of **java.util.Queue** API and **java.util.LinkedList** may be helpful in this problem.

## Input

The first line contains an integer **,** number of requests submitted.

The next **T** lines contains 2 type of requests that are submitted sequentially in time.

Requests are in one of the 2 types

|  |  |
| --- | --- |
| **Request** | **Interpretation** |
| MALE | A man submits a **date request.** |
| FEMALE | A woman submits a **date request** |

Two requests, even if of the same name, are submitted by two different people (i.e each person will only submit one request)

Each name is of length no more than 10 and consist of only lower-case and upper-case english letters

Users can have same names, and they will not affect logic of the app.

## Output

Whenever a match is made, print one line to the standard output of form “**A** matches **B**” where **A** is the name of male user, and **B** belongs to female one.

After all requests are handled, output a single line denoting the average waiting time of all **matched users.** Your answer should be rounded to 2 decimal places. If your answer is 2/3, you should print 0.67.

If number of **users** that get matched equals to zero, the average waiting time is 0.

## Examples

|  |  |
| --- | --- |
| **Input (dating1.in)** | **Output (dating1.out)** |
| 5  MALE Bob  MALE Daris  FEMALE Alice  MALE Bob  FEMALE Daris | Bob matches Alice  Daris matches Daris  1.25 |

|  |  |
| --- | --- |
| **Input (dating2.in)** | **Output (dating2.out)** |
| 8  FEMALE Bug  FEMALE Swift  MALE Nhat  FEMALE Bug  FEMALE Talor  MALE Bob  FEMALE Rose  MALE Minh | Nhat matches Bug  Bob matches Swift  Minh matches Bug  1.67 |

**For dating1 test case:**

|  |  |
| --- | --- |
| **Event** | **Explaination** |
| MALE Bob | Bob submits a request but there is no other woman waiting so he will wait for her. |
| MALE Daris | Daris submits a request, and he will wait with Bob. However, Bob is prioritized to be matched before Daris. |
| FEMALE Alice | Alice comes. Bob is the earliest man so the app will match Bob and Alice. Now, only Daris is waiting for his half. Bob has waited for 2 requests (one of Daris and one of Alice), hence his waiting time is 2. |
| MALE Bob | Another man named Bob (different from previous Bob) submits a date request. |
| FEMALE Daris | Daris the lady comes. Daris the gentleman submits the earliest request so two Daris are matched. |

|  |  |
| --- | --- |
| **Users** | **Wating time** |
| Bob (the first man) | 2 |
| Daris (male one) | 3 |
| Alice | 0 |
| Bob (second Bob) | No match ☹ |
| Daris (the lady) | 0 |

Hence, 4 users found their halves so the average waiting time is 1.25.

## Note:

1. A skeleton file has been given to help you. You should not create a new file or rename the file provided. You should develop your program using this skeleton file.
2. You are free to define your own helper methods and classes (or remove existing ones) if it is suitable but you must put all the new classes, if any, in the same skeleton file provided

## Skeleton File

You are given the skeleton file Dating.java. You should see the following contents when you open the file:

|  |
| --- |
| /\*\*  \* Name :  \* Matric. No :  \*/  import java.util.\*;  public class Dating {  private void run() {  }  public static void main(String args[]) {  Dating dating = new Dating();  dating.run();  }  } |