Classroom

Sharon is teaching a particular class. She loves her student and her students love her as a teacher. She is very passionate about teaching and she wants to help her students whenever she can. She is a very approachable teacher, not to mention friendly as well.

Today, Sharon is organizing a consultation session for all her students. She has prepared a consultation room where she will be waiting for her students to come. She has an open door policy: any student can come and go at any given time, and there are no limits to the number of students that can be present inside the room at any given time. Inside the room, there is a round table with many chairs surrounding it, enough to fit Sharon's whole class. Sharon will always be inside the room. When a student comes, the student will find an empty seat and sit there.

To monitor who is inside the room and who is sitting next to whom, Sharon has designed a simple query machine to manage the seating arrangement. This query machine is able to accept the following scenarios:

1. A new student enters the room, taking a specific seat.
2. A student leaves the room.
3. Sharon wants to list all students who are inside the room, starting from her in clockwise order.

As useful as this query machine can be, Sharon must prepare her teaching materials for the consultation session. Therefore, Sharon approached Ryan, her cousin, to help her code this simple query machine. Imagine that you are in Ryan's shoes and you are going to code this machine. Your task is to help Sharon build this query machine so that she can manage her students during the consultation.

Input

The first line of input contains an integer **N** (1 <= N <= 200), the number of queries. **N** rows follow. Each of the N rows contains a query for the machine to answer. The queries will follow the following specification:

Query Type Input Format: `<QUERY_TYPE> <APPROPRIATE_PARAMETERS>`

1. `enter STUDENT_1 STUDENT_2 k`
   The student STUDENT_1 enters the classroom. The rule is like this:

   - if k = 0, STUDENT_1 will sit next to STUDENT_2 (clockwise order)
   - if k = 1, STUDENT_1 will sit next to the student sitting next to STUDENT_2 (clockwise order)
   - if k = 2, STUDENT_1 will sit next to student sitting next to the student sitting next to STUDENT_2 (clockwise order)
   - and so on

   It is guaranteed that k is strictly less than the current number of students inside the classroom. It is also guaranteed that STUDENT_2 is inside the room when STUDENT_1 enters and STUDENT_1 is not yet inside the classroom when this query comes.

2. `leave STUDENT_NAME`
   The student with name STUDENT_NAME leaves the classroom. It is guaranteed that the student will be inside the classroom before this query comes.

3. `list`
   Print the name of all students who are currently at the table in clockwise order, starting from Sharon in one line. Each name is separated by a single space. There is no whitespace after the last student's name.

It is guaranteed that student names are all less than 20 characters and contains only English letters ('A-Z', 'a-z'). No two students have the same name.

Output
Print according to the specification of the queries given above. The last line of your output should contain a newline character.

Sample Input
```
7
enter Anthony Sharon 0
enter Diga Anthony 1
enter Evan Sharon 1
list
leave Evan
enter Nathan Anthony 2
list
```

Sample Output
```
Sharon Diga Evan Anthony
Sharon Diga Nathan Anthony
```

Skeleton
You are given the skeleton file Classroom.java (see contents on file).

Notes
1. You **must use linked list** to solve this problem.
2. You are free to define your own linked list class (encouraged for practice and skeleton file given), but you are allowed to use Java's built-in linked list implementation if it is suitable for this problem.
3. You are free to (and should) modify the skeleton file and add more attributes or methods when necessary.