2003/2004 SOUTHERN CALIFORNIA REGIONAL
ACM INTERNATIONAL COLLEGIATE PROGRAMMING CONTEST

Problem 0
Contest System Quality Assurance Tester

Write a program to score a small, three-problem programming contest. Each input line contains six space-separated integers representing raw score data. The first three integers are in the range 0..100000. They represent seconds taken to solve the first, second, and third problems, respectively. Zero seconds indicates that a problem has not been solved. The last three integers are in the range 0..100, representing the attempts taken to solve the first, second, and third problems, respectively. Every failed attempt is penalized with 20 minutes, but only for problems that are eventually solved.

Each output line should begin with the string "`team`", followed by a single space, the input line number, a colon, a single space, the number of solved problems, a comma, a single space, and the total number of seconds including penalties it took for the solved problems.

Sample Input

```
0 777 0 4 1 1
1 1 1 1 1 1
```

Output for the Sample Input

```
team 1: 1, 777
team 2: 3, 3
```

The warmup problem is at least as important to the judges as it is to the contestants. Contestants accustom themselves to the environment and the special commands, while the judges observe the first full-blown test with 60 or so teams of caffeine-pumped programmers beating up on the network. Please execute the special contest commands at least once to ensure proper functioning:

1. First, submit a question using the *question* command. Any question. Really. If you have no contest-related questions, we recommend "Who's buried in Grant's tomb?"
2. Use the *answers* command to view any questions already answered.
3. Next use the *score* command to see the scoreboard. Make sure that your team name and username are correct.
4. Now use the *timeleft* command. Please do *not* write a script that executes the timeleft command repeatedly. The judges use an accurate but highly radioactive atomic clock located directly beneath your workstation. Each execution of timeleft emits the radiation equivalent to one chest x-ray.

   Okay, just kidding about the radioactive clock thing, but the timeleft command does consume some of *your* precious computing resources. Your workstation clock is continually synced with the judges' clock. Simply note the contest end time from the timeleft command on an occasional basis.
5. Enter *getdata* to retrieve the sample input and sample output for the current problem set.
6. Write the team scoring program described above and use the *compile* command.
7. When the program passes your testing, use the *submit* command. **During the real contest, the judges don't recommend submitting untested programs, but for the warmup, submit your program, working or not.**
8. Wait a few minutes, and then issue the *status* command.