2004 Mid-Atlantic Regional Programming Contest
Practice Round

Welcome to the practice round for the 2004 Programming Contest. Before you start the contest, please be aware of the following notes:

1. There is one (1) practice problem. Please submit solutions or request clarifications **for this problem only.** Unless you have a real question about the problem, please submit at most one clarification request, and at most two runs. It is important that everyone have a chance to see how the system works.

2. After (or even before) completing the practice problem, please read all of the notes listed here. They are designed to help you solve the problems during the contest.

3. During the contest, a variety of reference materials for the supported compilers, libraries, and editors are available at

   `http://midatl.cs.vt.edu/contest/`

   The contest scoreboard is also accessible through that URL. Note that during the contest you may only access web pages through that link.

4. All solutions must read from standard input and write to standard output. In C this is scanf/printf, in C++ this is cin/cout, and in Java this is System.in/System.out. The judges will ignore all output sent to standard error. (You may wish to use standard error to output debugging information.) From your workstation you may test your program with an input file by redirecting input from a file:

   ```
   program < file.in
   ```

5. Solutions for problems submitted for judging are called runs. Each run will be judged. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first. DO NOT request clarifications on when a response will be returned. If you have not received a response for a run within 60 minutes of submitting it, **you may have a runner ask the site judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.**

   The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Correct	Your submission has been judged correct.
Incorrect Output	Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Incomplete Output	Your submission did not produce all of the required output.
Excessive Output	Your submission generated output in addition to or instead of what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve the judges' test data within 30 seconds.

6. A team's score is based on the number of problems they solve and penalty points, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty points.

7. This problem set contains sample input and output for each problem. However, you may be assured that the judges will test your submission against several other more complex datasets, which will not be revealed until after the contest. Your major challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive an incorrect judgment, you should consider what other datasets you could design to further evaluate your program.

8. In the event that you feel a problem statement is ambiguous, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient; no clarification is necessary." If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.

   Additionally, you may submit a clarification request asking for the correct output for input you provide. The judges will seek to respond to these requests with the correct output. These clarification requests will be answered only when no clarifications regarding ambiguity are pending. The judges reserve the right to suspend responding to these requests during the contest. If the provided input does not meet the specifications of the problem, or contains numbers that are not representable using the available programming languages, the response "illegal input" will be returned.

   If a clarification, including output for a given input, is issued during the contest, it will be broadcast to all teams.

9. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

10. All lines end with a newline character (`\n`, `endl`, or `println()`).

11. All input sets used by the judges will follow the input format specification found in the problem description.

12. Unless otherwise specified, all numbers will appear in the input and should be presented in the output beginning with the `-` if negative, followed immediately by 1 or more decimal digits. If it is a real number, then the decimal point appears, followed by any number of decimal digits (for output of real numbers the number of decimal digits will be specified in the problem description as the "precision"). All real numbers printed to a given precision should be rounded to the nearest value.

    In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure you use a printing technique that rounds to the appropriate precision.

13. Good luck, and HAVE FUN!!!

Practice Problem: A Contesting Decision

Judging a programming contest is hard work, with demanding contestants, tedious decisions, and monotonous work. Not to mention the nutritional problems of spending 12 hours with only donuts, pizza, and soda for food. Still, it can be a lot of fun.

Software that automates the judging process is a great help, but the notorious unreliability of some contest software makes people wish that something better were available. You are part of a group trying to develop better, open source, contest management software, based on the principle of modular design.

Your component is to be used for calculating the scores of programming contest teams and determining a winner. You will be given the results from several teams and must determine the winner.

Scoring

There are two components to a team's score. The first is the number of problems solved. The second is penalty points, which reflects the amount of time and incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved.

So if a team solved problem one on their second submission at twenty minutes, they are charged 40 penalty points. If they submit problem 2 three times, but do not solve it, they are charged no penalty points. If they submit problem 3 once and solve it at 120 minutes, they are charged 120 penalty points. Their total score is two problems solved with 160 penalty points.

The winner is the team that solves the most problems. If teams tie for solving the most problems, then the winner is the team with the fewest penalty points.

Input

For the programming contest your program is judging, there are four problems. You are guaranteed that the input will not result in a tie between teams after counting penalty points.

Line 1	`<nTeams>`
Line 2–$n+1$	`<Name> <p1Sub> <p1Time> <p2Sub> <p2Time> ... <p4Time>` The first element on the line is the team name, which contains no whitespace. Following that, for each of the four problems, is the number of times the team submitted a run for that problem and the time at which it was solved correctly (both integers). If a team did not solve a problem, the time will be zero. The number of submissions will be at least one if the problem was solved.

Output

The output consists of a single line listing the name of the team that won, the number of problems they solved, and their penalty points.

Example

Input:

```
4
Stars 2 20 5 0 4 190 3 220
Rockets 5 180 1 0 2 0 3 100
Penguins 1 15 3 120 1 300 4 0
Marsupials 9 0 3 100 2 220 3 80
```

Output:

```
Penguins 3 475
```