UCSD Fall 2007 Programming Contest

Sponsored by

The DiNI Group

Problem 1: Fun Projects

At an open-source fair held at a major university (UCSD of course), leaders of open-source projects put sign-up sheets on the wall, with the project name at the top in capital letters for identification.

Students then signed up for projects using their userids. A userid is a string of lower-case letters and numbers starting with a letter.

The organizer then took all the sheets off the wall and typed in the information.

Your job is to summarize the number of students who have signed up for each project. Some students were overly enthusiastic and put their name down several times for the same project. That's okay, but they should only count once. Students were asked to commit to a single project, so any student who has signed up for more than one project should not count for any project.

There are at most 10,000 students at the university, and at most 100 projects were advertised.

Input Specification:

The input contains several test cases, each one ending with a line that starts with the digit 1. The last test case is followed by a line starting with the digit 0.

Each test case consists of one or more project sheets. A project sheet consists of a line containing the project name in capital letters, followed by the userids of students, one per line.

Output Specification:

For each test case, output a summary of each project sheet. The summary is one line with the name of the project followed by the number of students who signed up. These lines should be printed in decreasing order of number of signups. If two or more projects have the same number of signups, they should be listed in alphabetical order.

Sample input

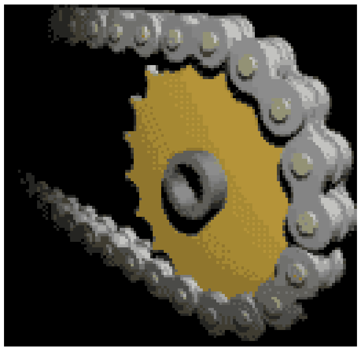
```
UBQTS TXT
tthumb
LIVESPACE BLOGJAM
philton
aeinstein
YOUBOOK
```

```
j97lee
sswxyzy
j97lee
aeinstein
SKINUX
1
0
```

Output for Sample Input

```
YOUBOOK 2
LIVESPACE BLOGJAM 1
UBQTS TXT 1
SKINUX 0
```

Problem 2: Tour de La Jolla



A racing bike is driven by a chain connecting two sprockets. Sprockets are grouped into two clusters: the front cluster (typically consisting of 2 or 3 sprockets) and the rear cluster (typically consisting of between 5 and 10 sprockets). At any time the chain connects one of the front sprockets to one of the rear sprockets. The drive ratio -- the ratio of the angular velocity of the pedals to that of the wheels -- is n:m where n is the number of teeth on the rear sprocket and m is the number of teeth on the front sprocket. Two drive ratios $d_1<d_2$ are adjacent if there is no other drive ratio $d_1<d_3<d_2$. The *spread* between a pair of drive ratios $d_1<d_2$ is their quotient: d_2/d_1. You are to compute the maximum spread between two adjacent drive ratios achieved by a particular pair of front and rear clusters.

Input consists of several test cases, followed by a line containing 0. Each test case is specified by the following input:

- f: the number of sprockets in the front cluster;
- r: the number of sprockets in the rear cluster;
- f integers, each giving the number of teeth on one of the gears in the front cluster;
- r integers, each giving the number of teeth on one of the gears in the rear cluster.

You may assume that no cluster has more than 10 sprockets and that no gear has fewer than 10 or more than 100 teeth.

For each test case, output the maximum spread rounded to two decimal places.

Sample Input

```
2 4
40 50
12 14 16 19
0
```

Output for Sample Input

```
1.19
```

Problem 3: Smart Exchange

Now that you are becoming financially savvy, you have decided to use your $1000 entrance scholarship to engage in currency speculation. So you consult your financial advisor, who tries to predict the closing exchange rate between Canadian and U.S. dollars for each of the next several days. On any given day, you can switch all of your money from Canadian to U.S. dollars, or vice versa, at the prevailing exchange rate, less a 3% commission, less any fraction of a cent.

Assuming that your advisor is correct, what's the maximum amount of money you can have, in Canadian dollars, when you're done?

The input contains a number of test cases, followed by a line containing 0. Each test case begins with $0 < d \leq 365$, the number of days that your advisor can predict. *d* lines follow, giving the price of a U.S. dollar in Canadian dollars, as a real number. For each test case, output a line giving the maximum amount of money, in Canadian dollars and cents, that it is possible to have at the end of the last prediction, assuming you may exchange money on any subset of the predicted days, in order.

Sample Input

```
3
1.0500
0.9300
0.9900
2
1.0500
1.1000
0
```

Output for Sample Input

```
1001.60
1000.00
```

Problem 4: Scicomp

Every October, the Kingdom of Scicomp holds a jousting tournament. In each of a series of event, a pair of knights attempt to knock each other from their respective horses. The winning knight is paired with another, while the loser is eliminated. This process continues until all but one knight is eliminated; this knight is declared champion.

The tournament schedule is organized so that no knight needs to compete in more than *e* events to be champion, for the minimum possible *e* given *k*, the number of knights. In order to construct the schedule, it may be necessary to identify several knights who compete in fewer than *e* events; these knights are said to be awarded a *bye* and are excluded from the first round of competition.

The first round of competition involves pairing as many knights as possible among those who are not awarded a bye. The competition is more interesting if the knights in each pair are as evenly matched in ability as possible. You are to determine which knights should be awarded a bye so as to make the first round as interesting as possible.

Standard input consists of several test cases followed by a line containing 0. Each test case begins with an integer $1 < k \leq 2500$, the number of knights. *i* lines follow, each giving the name and ability of a knight. The name is a string of lower case letters not longer than 20; the ability is a real number.

The *mismatch* between knights with abilities *a* and *b* respectively is defined to be $(a-b)^2$. For each test case, output the names of the knights to be given a bye such that the sum of all mismatch values for pairs of knights competing in the first round is minimized. If there are several solutions, any will do. Output an empty line between test cases.

Sample Input

```
3
gallahad 10
lancelot 11
mccartney 2
0
```

Output for Sample Input

```
Mccartney
```

Problem 5: CSEGangs

The downtown core of CSE City is laid out as a grid, with numbered streets running north-south from 1st Street in the west to 20th Street in the east, and numbered avenues running east-west from 1st Avenue in the north to 20th Avenue in the south. The area is controlled by two gangs, the CSGang and the CEGang. The boundary between their territory is the Green Line, running diagonally from the intersection of 1st Street and 1st Avenue to the intersection of 20th Street and 20th Avenue. The CSGang control the area to the southwest of the Green Line, and the CEGang the area to the northeast.

To prove their virility, the CSGang go on "runs" through CEGang's territory, starting at 1st Avenue and 1st Street and ending at a point on the Green Line that varies from night to night. A run may return to the Green Line in between but never crosses it. A run uses avenues only in the east direction and streets only in the south direction. Thus a run can be described by a string of E's and S's of length 2N-2; such a run ends at Nth Street and Nth Avenue.

The CSGang judge the runs made on a given night (all of which have the same length) by how "OG" they are. A run R1 is more OG than a run R2 if and only if:

- R2 returns to the Green Line for the first time at an earlier point than when R1 returns to the Green Line, or
- R1 and R2 return to the Green Line at the same point, but the portion of R1 to that point (ignoring the initial E and final S) is more OG than the portion of R2 to that point (also ignoring the initial E and final S), or
- R1 and R2 return to the Green Line at the same point and are identical to that point, but the rest of R1 is more OG than the rest of R2.

Examples corresponding to these three cases:

- EESS is more OG than ESES.
- EEESSS is more OG than EESESS
- EESSEESS is more OG than EESSESES.

If all the runs of a given length are ordered according to how OG they are, then the rank of a run is its position in the resulting list. EESS has rank 1 and ESES has rank 2.

Your task is to write a program to help the CSGangs plan and judge their nightly activities. The input to the program is a series of instances followed by 0 0. An instance consists of a line containing a positive integer N, representing the terminus of that night's run (Nth Street and Nth Avenue), followed by positive integer M. The output corresponding to each instance is the run of length 2N-2 of rank M, or ERROR if there are fewer than M runs of length 2N-2.

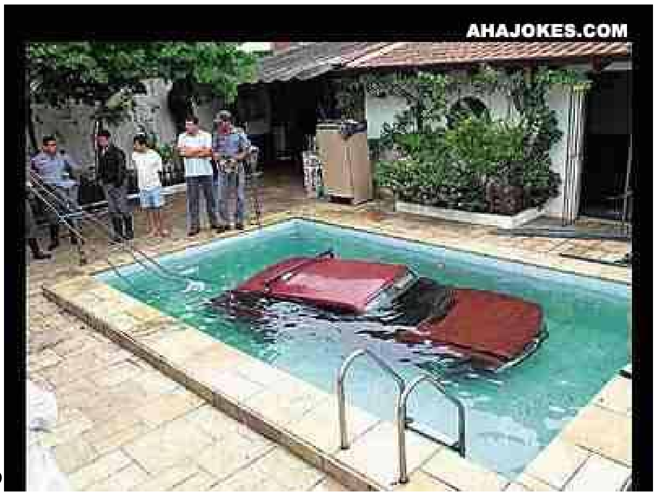
Sample Input

```
3 1
3 2
3 3
0 0
```

Output for Sample Input

```
EESS
ESES
ERROR
```

Problem 6: Carpool



You and your friends have just completed your CS assignments, and because of the nice weather, decide to go to Paul's house for a BBQ. Unfortunately, after all that coding, you all are too tired to walk. Fortunately, there are enough cars to take everyone.

Paul remembers that he needs to stop off at the supermarket along the way to buy some burgers and soda.

Jenny proposes that they stop at her house to get a frisbee.

Jim decides that he doesn't like burgers, and wants to grab a pizza along the way.

After having spent so long in the computer lab, Jake's eyes have become very sensitive to sunlight, so he needs to stop at his house for his sunglasses.

And so it goes: each person needs to run a little errand along the way. At this rate, your friends worry that it will be dark by the time they get to Paul's house. They launch into a heated discussion to about who should go in which car to minimize the time needed for everyone to reach Paul's house. The discussion itself, of course, wastes precious time that could be better spent at the BBQ.

To help the group, you will write a program to settle the discussion by computing an optimal assignment of people to cars. The overall travel time is the maximum of the travel times of each car. An optimal assignment is one that minimizes the overall travel time.

Your program will be provided with a representation of the roads in the city, in the form of distances between major landmarks. Assume that every car always travels at 60 miles per hour (one mile per minute). Each stop (at a supermarket, someone's house, etc.) takes five minutes.

Although your friends have many cars between them, to be nice to the environment, they decide to take no more cars than necessary. Each car can take at most five people.

Input Specification:

The first line of input contains two integers n and m, 1 ≤ n ≤ 15, 1 ≤ m ≤ 1000, the number of people in the group and the number of roads in the city. The places that must be visited along the way are numbered 1 through n. The campus is numbered 0, and Paul's house is numbered n+1. An additional m lines follow, each containing three integers describing a stretch of road. The first two integers are in the range 0 to n+1 inclusive, and describe the two places connected by the stretch of road. The third integer specifies the length of the stretch of road, in miles. A road may be taken in both directions. There is a sequence of roads connecting every place in the city to every other place.

Output Specification:

Output a single integer, giving the number of minutes required for everyone to reach Paul's house using an optimal assignment of people to cars.

Sample Input:

```
1 2
0 1 15
1 2 10
```

Output for Sample Input:

```
30
```