UW Madison's

2005 ACM-ICPC Individual Placement Test

September 18, 1:00-6:00pm, 1350 CS

Overview:
This test consists of eight problems, which will be referred to by the following names (respective of order):

shoe, babylon, chandelier, numbers, knights, department, land, **fire.**

Please note that some problems also include a problem number (or letter). Please ignore these labelings and use the names above.

Input/Output:
Your programs should take input from standard in (i.e. the keyboard) and output to standard out (i.e. the terminal). As is standard practice for the ICPC, you may assume that the input strictly follows the description in the problems. If a problem description does not indicate how the input will be terminated, you may assume it will be with the EOF character. It is your responsibility to ensure that your output **precisely** matches that described in the problems, or else risk your program being rejected with a "Presentation Error".

Problem Submission:
To submit a program, send e-mail to mwa+icpc@cs.wisc.edu and attach your source code. The subject line should contain only the problem name, **prob,** that you are submitting, and the attached source code should be named **prob**.{c|cpp|java}. For example, if you are using C++ and submitting the problem "land", the file should be named "land.cpp". You will receive the results of your submission as soon as possible via e-mail.

Clarifications:
As in the ICPC, you may submit clarification requests as well. They should be sent to mwa+icpc@cs.wisc.edu, with a subject of "Clarification-**prob**", where **prob** is the name of the problem you wish to be clarified. Replace **prob** with "general" if there is an issue with the contest as a whole. Accepted clarification requests will be answered to all those taking the test via e-mail. Experience of previous years learns that most clarification requests are rejected, and receive a simple response such as "Read the problem description".

Printing:
You may print to the printer at any time during the test.

After the Test:
The proctor will announce when time is up. Please stop working at this time and take a moment to fill out the form on the back of this sheet and turn it in to the proctor (you may keep the problems). You are invited to join us for pizza and soda after the test in 1325 CS.

Final Submissions:
You will have until 11:59 pm on Tuesday to make **one** final submission of any problems which you did not successfully solve during the test period. Please note that only your **first** submission after 6:00 pm on Sunday will be evaluated in case multiple are submitted. **Due to this extra time allowance, please refrain from discussing any of the problems after the test.**

Information Form:

Name:

CS Login:

Student status (i.e. Junior, first year grad student):

Year of birth:_____ , Year starting college:_____

What do you feel are your strengths with respect to the ICPC?

What programming languages do you prefer?

Is there anyone that you would prefer to be placed on a team with (if possible)?

How did you hear about the ICPC (e-mail, flier, word of mouth, etc.):

From Skiena and Revilla, *Programming Challenges: The Programming Contest Training Manual*
Springer-Verlag, New York, 2003. ISBN: 0-387-00163-8.

110405 Shoes maker's problem

A shoemaker has N orders from customers which he must satisfy. The shoemaker can work on only one job in each day, and jobs usually take several days. For the ith job, the integer T_i $(1 \leq T_i \leq 1,000)$ denotes the number of days it takes the shoemaker to finish the job.

But popularity has its price. For each day of delay before starting to work on the ith job, the shoemaker has agreed to pay a fine of S_i $(1 \leq S_i \leq 10,000)$ cents per day. Help the shoemaker by writing a program to find the sequence of jobs with minimum total fine.

Input

The input begins with a single positive integer on a line by itself indicating the number of the test cases, followed by a blank line. There is also a blank line between two consecutive cases.

The first line of each case contains an integer reporting the number of jobs N, where $1 \leq N \leq 1,000$. The ith subsequent line contains the completion time T_i and daily penalty S_i for the ith job.

Output

For each test case, your program should print the sequence of jobs with minimal fine. Each job should be represented by its position in the input. All integers should be placed on only one output line and each pair separated by one space. If multiple solutions are possible, print the first one in lexicographic order.

The output of two consecutive cases must be separated by a blank line.

Sample Input

```
1

4
3 4
1 1000
2 2
5 5
```

Sample Output

```
2 1 3 4
```


The Tower of Babylon

Perhaps you have heard of the legend of the Tower of Babylon. Nowadays many details of this tale have been forgotten. So now, in line with the educational nature of this contest, we will tell you the whole story:

The babylonians had *n* types of blocks, and an unlimited supply of blocks of each type. Each type−*i* block was a rectangular solid with linear dimensions (x_i, y_i, z_i). A block could be reoriented so that any two of its three dimensions determined the dimensions of the base and the other dimension was the height. They wanted to construct the tallest tower possible by stacking blocks. The problem was that, in building a tower, one block could only be placed on top of another block as long as the two base dimensions of the upper block were both strictly smaller than the corresponding base dimensions of the lower block. This meant, for example, that blocks oriented to have equal−sized bases couldn't be stacked.

Your job is to write a program that determines the height of the tallest tower the babylonians can build with a given set of blocks.

Input and Output

The input file will contain one or more test cases. The first line of each test case contains an integer *n*, representing the number of different blocks in the following data set. The maximum value for *n* is 30. Each of the next *n* lines contains three integers representing the values x_i, y_i and z_i.

Input is terminated by a value of zero (0) for *n*.

For each test case, print one line containing the case number (they are numbered sequentially starting from 1) and the height of the tallest possible tower in the format "`Case` *case*`: maximum height =` *height*"

Sample Input

```
1
10 20 30
2
6 8 10
5 5 5
7
1 1 1
2 2 2
3 3 3
4 4 4
5 5 5
6 6 6
7 7 7
5
31 41 59
26 53 58
97 93 23
84 62 64
33 83 27
0
```

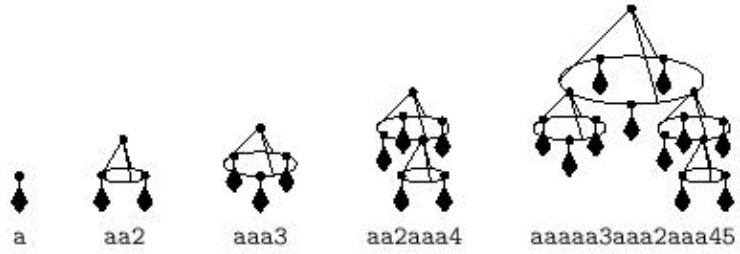
Sample Output

```
Case 1: maximum height = 40
Case 2: maximum height = 21
Case 3: maximum height = 28
Case 4: maximum height = 342
```

3215 – Chandelier

Europe – Northeastern Europe & Russian Republic – 2004/2005

Lamps–O–Matic company assembles very large chandeliers. A chandelier consists of multiple levels. On the first level crystal pendants are attached to the rings. Assembled rings and new pendants are attached to the rings of the next level, and so on. At the end there is a single large ring –– the complete chandelier with multiple smaller rings and pendants hanging from it. A special–purpose robot assembles chandeliers. It has a supply of crystal pendants and empty rings, and a stack to store elements of a chandelier during assembly. Initially the stack is empty. Robot executes a list of commands to assemble a chandelier.



On command ``a'' robot takes a new crystal pendant and places it on the top of the stack. On command ``1'' to ``9'' robot takes the corresponding number of items from the top of the stack and consecutively attaches them to the new ring. The newly assembled ring is then placed on the top of the stack. At the end of the program there is a single item on the stack –– the complete chandelier. Unfortunately, for some programs it turns out that the stack during their execution needs to store too many items at some moments. Your task is to optimize the given program, so that the overall design of the respective chandelier remains the same, but the maximal number of items on the stack during the execution is minimal. A pendant or any complex multi–level assembled ring count as a single item of the stack. The design of a chandelier is considered to be the same if each ring contains the same items in the same order. Since rings are circular it does not matter what item is on the top of the stack when the robot receives a command to assemble a new ring, but the relative order of the items on the stack is important. For example, if the robot receives command ``4'' when items $\langle i_1, i_2, i_3, i_4 \rangle$ being the topmost), then the same ring is also assembled if these are on the top of the stack in this order (i_1 items are arranged on the stack in the following ways: $\langle i_2, i_3, i_4, i_1 \rangle$, or $\langle i_3, i_4, i_1, i_2 \rangle$, or $\langle i_4, i_1, i_2, i_3 \rangle$.

Input

Input file contains several test cases. Each of them consists of a single line with a valid program for the robot. The program consists of at most 10 000 characters.

Output

For each test case, print two output lines. On the first line write the minimal required stack capacity (number of items it can hold) to assemble the chandelier. On the second line write some program for the assembly robot that uses stack of this capacity and results in the same chandelier.

Sample Input

```
aaaaa3aaa2aaa45
```

Sample Output

```
6
aaa3aaa2aaa4aa5
```

Northeastern Europe & Russian Republic 2004–2005

2687 – Amusing Numbers

Europe – Northeastern – 2002/2003

Let us consider the set of integer numbers between 1 and N inclusive. Let us order them lexicographically (i. e. like in the vocabulary), for example, for $N = 11$ the order would be: 1, 10, 11, 2, 3, 4, 5, 6, 7, 8, 9.

Let us denote the position of the number K in this ordering as $Q_{N, K}$. For example, $Q_{11, 2} = 4$. Given numbers K and M find the smallest N such that $Q_{N, K} = M$.

Input

Input file contains several test cases, one per line. Each of them consists of two integer numbers K and M ($1 \leq K, M \leq 10^9$) separated by a space.

Output

For each input case write a different output line. If such N that $Q_{N, K} = M$ exists then write the smallest such N, otherwise write `0'.

Sample Input

```
2 4
2 1
100000001 1000000000
1000000000 11
```

Sample Output

```
11
0
100000000888888879
0
```

Northeastern 2002–2003

From Skiena and Revilla, *Programming Challenges: The Programming Contest Training Manual*
Springer-Verlag, New York, 2003. ISBN: 0-387-00163-8.

111303 The Knights Of The Round Table

King Arthur is planning to build the round table in a room which has a triangular window in the ceiling. He wants the sun to shine on his round table. In particular, he wants the table to be totally in the sunlight when the sun is directly overhead at noon.

Thus the table must be built in a particular triangular region of the room. Of course, the king wants to build the largest possible table under the circumstances.

As Merlin is out to lunch, write a program which finds the radius of the largest circular table that fits in the sunlit area.

Input

There will be an arbitrary number of test cases, each represented by three real numbers (a, b, and c), which stand for the side lengths of the triangular region. No side length will be greater than 1,000,000, and you may assume that $\max(a, b, c) \leq (a + b + c)/2$.

You must read until you reach the end of the file.

Output

For each room configuration read, you must print the following line:

`The radius of the round table is:` r

where r is the radius of the largest round table that fits in the sunlit area, rounded to three decimal digits.

Sample Input

```
12.0 12.0 8.0
```

Sample Output

```
The radius of the round table is: 2.828
```


ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem E: Department

Input file: `dept.in`
Output file: `dept.out`
Program file: `dept.pas` or `dept.cpp`

The Department of Security has a new headquarters building. The building has several floors, and on each floor there are rooms numbered $xxyy$ where yy stands for the room number and xx for the floor number, $0 < xx, yy \leq 10$. The building has 'pater-noster' elevator, i.e. elevator build up from several cabins running all around. From time to time the agents must visit the headquarters. During their visit they want to visit several rooms and in each room they want to stay for some time. Due to the security reasons, there can be only one agent in the same room at the same time, The same rule applies to the elevators. The visits are planned in the way ensuring they can be accomplished within one day. Each agent visits the headquarters at most once a day.

Each agent enters the building at the 1st floor, passes the reception and then starts to visit the rooms according to his/her list. Agents always visit the rooms by the increasing room numbers. The agents form a linear hierarchy according to which they have assigned their one letter personal codes. The agents with higher seniority have lexicographically smaller codes. No two agents have the same code.

If more then one agent want to enter a room, or an elevator, the agents have to form a queue. In each queue, they always stand according to their codes. The higher the seniority of the agent, the closer to the top of the queue he stands. Every 5 s (seconds) the first agent in the queue in front of the elevator enters the elevator. After visiting the last room in the headquarters each agent uses if necessary elevator to the first floor and exits the building.

The times necessary to move from a certain point in the headquarters to another are set as follows: Entering the building, i.e. passing the reception and reaching the elevator, or a room on the first floor takes 30 s. Exiting the building, i.e. stepping out of the elevator or a room on the first floor and passing the reception takes also 30 s. On the same floor, the transfer from the elevator to the room (or to the queue in front of the room), or from the room to the elevator (or to the queue in front of the elevator), or from one room to another (or to the queue in front of the room) takes 10 s. The transfer from one floor to the next floor above or below in an elevator takes 30 s. Write a program that determines time course of agent's visits in the headquarters.

Input

The input file contains the descriptions of $n \geq 0$ visits of different agents. The first line of the description of each visit consists of agent's one character code C, $C =$ `A`, ..., `Z`, and the time when the agent enters the headquarters. The time is in the format HH:MM:SS (hours, minutes, seconds). The next lines (there will be at least one) contain the room number, and the length of time intended to stay in the room, time is in seconds. Each room is in a separate line. The list of rooms is sorted according to the increasing room number. The list of rooms ends by the line containing 0. The list of the descriptions of visits ends by the line containing the character dot.

Output

The output contains detailed records of each agent's visit in the headquarters. For each agent, there will be a block. Blocks are ordered in the order of increasing agent's codes. Blocks are separated by an empty line. After the last block there is an empty line too. The first line of a block contains the code of agent. Next lines contain the starting and ending time (in format HH:MM:SS) and the descriptions of his/her activity. Time data will be separated by one blank character. Description will be separated from time by one blank character. Description will have a form `Entry`, `Exit` or Message. The Message can be one of the following: `Waiting in elevator queue`, `Waiting in front of room` RoomNumber, `Transfer from room` RoomNumber `to room` RoomNumber, `Transfer from elevator to room` RoomNumber, `transfer from` RoomNumber `to elevator`, `Stay in room` RoomNumber, `Stay in elevator`.

Example

Input file

```
A 10:00:00
0101 100
0110 50
0202 90
0205 50
0
B 10:01:00
0105 100
0201 5
0205 200
0
.
```

Output file

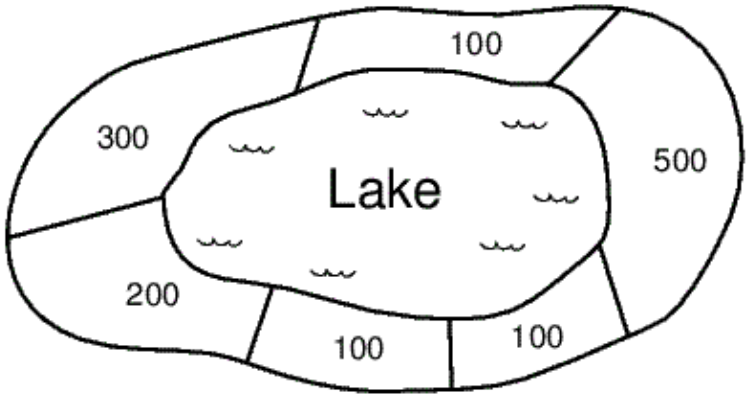
```
A
10:00:00 10:00:30 Entry
10:00:30 10:02:10 Stay in room 0101
10:02:10 10:02:20 Transfer from room 0101 to room 0110
10:02:20 10:03:10 Stay in room 0110
10:03:10 10:03:20 Transfer from room 0110 to elevator
10:03:20 10:03:50 Stay in elevator
10:03:50 10:04:00 Transfer from elevator to room 0202
10:04:00 10:05:30 Stay in room 0202
10:05:30 10:05:40 Transfer from room 0202 to room 0205
10:05:40 10:07:40 Waiting in front of room 0205
10:07:40 10:08:30 Stay in room 0205
10:08:30 10:08:40 Transfer from room 0205 to elevator
10:08:40 10:09:10 Stay in elevator
10:09:10 10:09:40 Exit

B
10:01:00 10:01:30 Entry
10:01:30 10:03:10 Stay in room 0105
10:03:10 10:03:20 Transfer from room 0105 to elevator
10:03:20 10:03:25 Waiting in elevator queue
10:03:25 10:03:55 Stay in elevator
10:03:55 10:04:05 Transfer from elevator to room 0201
10:04:05 10:04:10 Stay in room 0201
10:04:10 10:04:20 Transfer from room 0201 to room 0205
10:04:20 10:07:40 Stay in room 0205
10:07:40 10:07:50 Transfer from room 0205 to elevator
10:07:50 10:08:20 Stay in elevator
10:08:20 10:08:50 Exit
```

3151 – Land Division Tax

Latin America – South America – 2004/2005

International Concrete Projects Company (ICPC) is a construction company which specializes in building houses for the high−end market. ICPC is planning a housing development for new homes around a lake. The houses will be built in lots of different sizes, but all lots will be on the lake shore. Additionally, every lot will have exactly two neighbors in the housing development: one to the left and one to the right.



Development plan indicating the sizes of the lots (in units of area) in the new housing development.

ICPC owns the land around the lake and needs to divide it into lots according to the housing development plan. However, the County Council has a curious regulation regarding land tax, intended to discourage the creation of small lots:

1. land can only be divided using a sequence of land divisions;
2. a land division is an operation that divides one piece of land into two pieces of land; and
3. for each land division, a land division tax must be paid.

Denoting by A the area of the largest resulting part of the division, the value of the land division tax is $A \times F$, where F is the division tax factor set yearly by the County Council. Note that due to (2), in order to divide a piece of land into N lots, $N - 1$ land divisions must be performed, and therefore $N - 1$ payments must be made to the County Council.

For example, considering the figure above, if the division tax factor is 2.5 and the first land division separates the lot of 500 units of area from the other lots, the land division tax to be paid for this first division is $2.5 \times (300 + 200 + 100 + 100 + 100)$. If the next land division separates the lot of 300 units together with its neighbor lot of 100 units, from the set of the remaining lots, an additional $2.5 \times (300 + 100)$ must be paid in taxes, and so on. Note also that some land divisions are not possible, due to (2). For example, after the first land division mentioned above, it is not possible to make a land division to separate the lot of 300 units together with the lot of 200 units from the remaining three lots, because more than two parts would result from that operation.

Given the areas of all lots around the lake and the current value of the division tax factor, you must write a program to determine the smallest total land division tax that should be paid to divide the land according to

the housing development plan.

Input

The input contains several test cases. The first line of a test case contains an integer N and a real F, indicating respectively the number of lots (1 ≤ N ≤ 200) and the land division tax factor (with precision of two decimal digits, 0 < F ≤ 5.00). The second line of a test case contains N integers X_i, representing the areas of contiguous lots in the development plan (0 < X_i ≤ 500, for 1 ≤ i ≤ N); furthermore, X_k is neighbour to X_{k+1} for 1 ≤ k ≤ N − 1, and X_N is neighbour to X_1. The end of input is indicated by N = F = 0.

Output

For each test case in the input your program must produce a single line of output, containing the minimum total land division tax, as a real number with precision of two decimal digits.

Sample Input

```
4 1.50
2 1 4 1
6 2.50
300 100 500 100 100 200
0 0
```

Sample Output

```
13.50
4500.00
```

South America 2004–2005

Problem A: Fire Station

A city is served by a number of fire stations. Some residents have complained that the distance from their houses to the nearest station is too far, so a new station is to be built. You are to choose the location of the fire station so as to reduce the distance to the nearest station from the houses of the disgruntled residents.

The city has up to 500 intersections, connected by road segments of various lengths. No more than 20 road segments intersect at a given intersection. The location of houses and firestations alike are considered to be at intersections (the travel distance from the intersection to the actual building can be discounted). Furthermore, we assume that there is at least one house associated with every intersection. There may be more than one firestation per intersection.

The Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of input contains two positive integers: *f*,the number of existing fire stations (*f* <= 100) and *i*, the number of intersections (*i* <= 500). The intersections are numbered from 1 to *i* consecutively. *f* lines follow; each contains the intersection number at which an existing fire station is found. A number of lines follow, each containing three positive integers: the number of an intersection, the number of a different intersection, and the length of the road segment connecting the intersections. All road segments are two−way (at least as far as fire engines are concerned), and there will exist a route between any pair of intersections.

The Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

You are to output a single integer: the lowest intersection number at which a new fire station should be built so as to minimize the maximum distance from any intersection to the nearest fire station.

Sample Input

```
1

1 6
2
1 2 10
2 3 10
3 4 10
4 5 10
5 6 10
6 1 10
```

Output for Sample Input

```
5
```