ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem A: Maya Calendar

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

During his last sabbatical, professor M. A. Ya made a surprising discovery about the old Maya calendar. From an old knotted message, professor discovered that the Maya civilization used a 365 day long year, called *Haab*, which had 19 months. Each of the first 18 months was 20 days long, and the names of the months were *pop, no, zip, zotz, tzec, xul, yoxkin, mol, chen, yax, zac, ceh, mac, kankin, muan, pax, koyab, cumhu*. Instead of having names, the days of the months were denoted by numbers starting from 0 to 19. The last month of Haab was called *uayet* and had 5 days denoted by numbers 0, 1, 2, 3, 4. The Maya believed that this month was unlucky, the court of justice was not in session, the trade stopped, people did not even sweep the floor.

For religious purposes, the Maya used another calendar in which the year was called *Tzolkin* (holly year). The year was divided into thirteen periods, each 20 days long. Each day was denoted by a pair consisting of a number and the name of the day. They used 20 names: *imix, ik, akbal, kan, chicchan, cimi, manik, lamat, muluk, ok, chuen, eb, ben, ix, mem, cib, caban, eznab, canac, ahau* and 13 numbers; both in cycles.

Notice that each day has an unambiguous description. For example, at the beginning of the year the days were described as follows:

1 imix, 2 ik, 3 akbal, 4 kan, 5 chicchan, 6 cimi, 7 manik, 8 lamat, 9 muluk, 10 ok, 11 chuen, 12 eb, 13 ben, 1 ix, 2 mem, 3 cib, 4 caban, 5 eznab, 6 canac, 7 ahau, and again in the next period *8 imix, 9 ik, 10 akbal* . . .

Years (both Haab and Tzolkin) were denoted by numbers 0, 1, . . . , where the number 0 was the beginning of the world. Thus, the first day was:

Haab: 0. pop 0
Tzolkin: 1 imix 0

Help professor M. A. Ya and write a program for him to convert the dates from the Haab calendar to the Tzolkin calendar.

Input

The date in Haab is given in the following format:
NumberOfTheDay. Month Year
The first line of the input file contains the number of the input dates in the file. The next n lines contain n dates in the Haab calendar format, each in separate line. The year is smaller then 5000.

Output

The date in Tzolkin should be in the following format:
Number NameOfTheDay Year
The first line of the output file contains the number of the output dates. In the next n lines, there are dates in the Tzolkin calendar format, in the order corresponding to the input dates.

Example

Input file:
```
3
10. zac 0
0. pop 0
10. zac 1995
```
Output file:
```
3
3 chuen 0
1 imix 0
9 cimi 2801
```

ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem B: Transportation

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

Ruratania is just entering capitalism and is establishing new enterprising activities in many fields including transport. The transportation company TransRuratania is starting a new express train from city A to city B with several stops in the stations on the way. The stations are successively numbered, city A station has number 0, city B station number m. The company runs an experiment in order to improve passenger transportation capacity and thus to increase its earnings. The train has a maximum capacity n passengers. The price of the train ticket is equal to the number of stops (stations) between the starting station and the destination station (including the destination station). Before the train starts its route from the city A, ticket orders are collected from all onroute stations. The ticket order from the station S means all reservations of tickets from S to a fixed destination station. In case the company cannot accept all orders because of the passenger capacity limitations, its rejection policy is that it either completely accept or completely reject single orders from single stations.

Write a program which for the given list of orders from single stations on the way from A to B determines the biggest possible total earning of the TransRuratania company. The earning from one accepted order is the product of the number of passengers included in the order and the price of their train tickets. The total earning is the sum of the earnings from all accepted orders.

Input

The input file is divided into blocks. The first line in each block contains three integers: passenger capacity n of the train, the number of the city B station and the number of ticket orders from all stations. The next lines contain the ticket orders. Each ticket order consists of three integers: starting station, destination station, number of passengers. In one block there can be maximum 22 orders. The number of the city B station will be at most 7. The block where all three numbers in the first line are equal to zero denotes the end of the input file.

Output

The output file consists of lines corresponding to the blocks of the input file except the terminating block. Each such line contains the biggest possible total earning.

Example

input file:
```
10 3 4
0 2 1
1 3 5
1 2 7
2 3 10
10 5 4
3 5 10
2 4 9
0 2 5
2 5 8
0 0 0
```
output file:
```
19
34
```

ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem C: John's trip

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

Little Johnny has got a new car. He decided to drive around the town to visit his friends. Johnny wanted to visit all his friends, but there was many of them. In each street he had one friend. He started thinking how to make his trip as short as possible. Very soon he realized that the best way to do it was to travel through each street of town only once. Naturally, he wanted to finish his trip at the same place he started, at his parents' house.

The streets in Johnny's town were named by integer numbers from 1 to n, $n < 1995$. The junctions were independently named by integer numbers from 1 to m, $m \leq 44$. No junction connects more than 44 streets. All junctions in the town had different numbers. Each street was connecting exactly two junctions. No two streets in the town had the same number. He immediately started to plan his round trip. If there was more than one such round trip, he would have chosen the one which, when written down as a sequence of street numbers is lexicographically the smallest. But Johnny was not able to find even one such round trip.

Help Johnny and write a program which finds the desired shortest round trip. If the round trip does not exist the program should write a message. Assume that Johnny lives at the junction ending the street No. 1 with smaller number. All streets in the town are two way. There exists a way from each street to another street in the town. The streets in the town are very narrow and there is no possibility to turn back the car once he is in the street.

Input

Input file consists of several blocks. Each block describes one town. Each line in the block contains three integers x, y, z, where $x > 0$ and $y > 0$ are the numbers of junctions which are connected by the street number z. The end of the block is marked by the line containing $x = y = 0$. At the end of the input file there is an empty block, $x = y = 0$.

Output

The output file consists of 2 line blocks corresponding to the blocks of the input file. The first line of each block contains the sequence of street numbers (single members of the sequence are separated by space) describing Johnny's round trip. If the round trip cannot be found the corresponding output block contains the message `Round trip does not exist`. The second line of each block is empty.

Example

input file:

```
1 2 1
2 3 2
3 1 6
1 2 5
2 3 3
3 1 4
0 0
1 2 1
2 3 2
1 3 3
2 4 4
0 0
0 0
```

output file:

```
1 2 3 5 4 6

Round trip does not exist
```

ACM International Collegiate Programming Contest 95/96

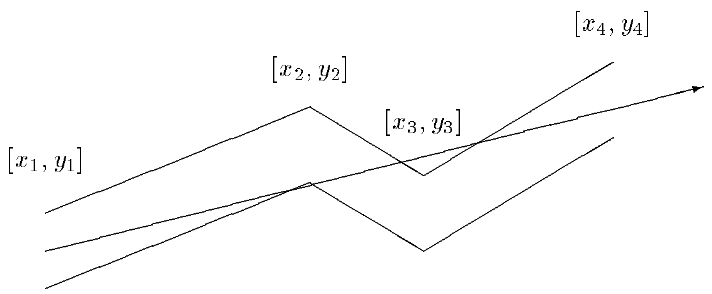
Sponsored by Microsoft

Central European Regional Contest

Problem D: Pipe

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

The GX Light Pipeline Company started to prepare bent pipes for the new transgalactic light pipeline. During the design phase of the new pipe shape the company ran into the problem of determining how far the light can reach inside each component of the pipe. Note that the material which the pipe is made from is not transparent and not light reflecting.



Each pipe component consists of many straight pipes connected tightly together. For the programming purposes, the company developed the description of each component as a sequence of points $[x_1, y_1]$, $[x_2, y_2]$, ..., $[x_n, y_n]$, where $x_1 < x_2 < \ldots x_n$. These are the upper points of the pipe contour. The bottom points of the pipe contour consist of points with y-coordinate decreased by 1. To each upper point $[x_i, y_i]$ there is a corresponding bottom point $[x_i, y_i - 1]$ (see picture above). The company wants to find, for each pipe component, the point with maximal x-coordinate that the light will reach. The light is emitted by a segment source with endpoints $[x_1, y_1 - 1]$ and $[x_1, y_1]$ (endpoints are emitting light too). Assume that the light is not bent at the pipe bent points and the bent points do not stop the light beam.

Input

The input file contains several blocks each describing one pipe component. Each block starts with the number of bent points $2 \le n \le 20$ on separate line. Each of the next n lines contains a pair of real values x_i, y_i separated by space. The last block is denoted with $n = 0$.

Output

The output file contains lines corresponding to blocks in input file. To each block in the input file there is one line in the output file. Each such line contains either a real value, written with precision of two decimal places, or the message `Through all the pipe.`. The real value is the desired maximal x-coordinate of the point where the light can reach from the source for corresponding pipe component. If this value equals to x_n, then the message `Through all the pipe.` will appear in the output file.

Example

Input file

```
4
0 1
2 2
4 1
6 4
6
0 1
```

```
2 -0.6
5 -4.45
7 -5.57
12 -10.8
17 -16.55
0
```

Output file

```
4.67
Through all the pipe.
```

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
```

ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem F: Joseph

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

The Joseph's problem is notoriously known. For those who are not familiar with the original problem: from among n people, numbered 1, 2, . . ., n, standing in circle every mth is going to be executed and only the life of the last remaining person will be saved. Joseph was smart enough to choose the position of the last remaining person, thus saving his life to give us the message about the incident. For example when $n = 6$ and $m = 5$ then the people will be executed in the order 5, 4, 6, 2, 3 and 1 will be saved.

Suppose that there are k good guys and k bad guys. In the circle the first k are good guys and the last k bad guys. You have to determine such minimal m that all the bad guys will be executed before the first good guy.

Input

The input file consists of separate lines containing k. The last line in the input file contains 0. You can suppose that $0 < k < 14$.

Output

The output file will consist of separate lines containing m corresponding to k in the input file.

Example

Input file:

```
3
4
0
```

Output file:

```
5
30
```

ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem G: Cipher

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

Bob and Alice started to use a brand-new encoding scheme. Surprisingly it is not a Public Key Cryptosystem, but their encoding and decoding is based on secret keys. They chose the secret key at their last meeting in Philadelphia on February 16th, 1996. They chose as a secret key a sequence of n distinct integers, $a_1, \ldots, a_n$, greater than zero and less or equal to n. The encoding is based on the following principle. The message is written down below the key, so that characters in the message and numbers in the key are correspondingly aligned. Character in the message at the position i is written in the encoded message at the position a_i, where a_i is the corresponding number in the key. And then the encoded message is encoded in the same way. This process is repeated k times. After kth encoding they exchange their message.

The length of the message is always less or equal than n. If the message is shorter than n, then spaces are added to the end of the message to get the message with the length n.

Help Alice and Bob and write program which reads the key and then a sequence of pairs consisting of k and message to be encoded k times and produces a list of encoded messages.

Input

The input file consists of several blocks. Each block has a number $0 < n \leq 200$ in the first line. The next line contains a sequence of n numbers pairwise distinct and each greater than zero and less or equal than n. Next lines contain integer number k and one message of ascii characters separated by one space. The lines are ended with eol, this eol does not belong to the message. The block ends with the separate line with the number 0. After the last block there is in separate line the number 0.

Output

Output is divided into blocks corresponding to the input blocks. Each block contains the encoded input messages in the same order as in input file. Each encoded message in the output file has the lenght n. After each block there is one empty line.

Example

Input file:

```
10
4 5 3 7 2 8 1 6 10 9
1 Hello Bob
1995 CERC
0
0
```

Output file:

```
BolHeol  b
C RCE
```

ACM International Collegiate Programming Contest 95/96

Sponsored by Microsoft

Central European Regional Contest

Problem H: Sticks

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

George took sticks of the same length and cut them randomly until all parts became at most 50 units long. Now he wants to return sticks to the original state, but he forgot how many sticks he had originally and how long they were originally. Please help him and design a program which computes the smallest possible original length of those sticks. All lengths expressed in units are integers greater than zero.

Input

The input file contains blocks of 2 lines. The first line contains the number of sticks parts after cutting. The second line contains the lengths of those parts separated by the space. The last line of the file contains zero.

Output

The output file contains the smallest possible length of original sticks, one per line.

Example

Input file:

```
9
5 2 1 5 2 1 5 2 1
4
1 2 3 4
0
```

Output file:

```
6
5
```