Problem A

Add All

Given a set of numbers your goal is to add them all, while minimizing the cost of the addition. At any point, you may choose to add any two of the integers. The cost of doing so is simply the sum of the two numbers. For example, the cost of adding 1 and 10 is 11. 3, 6 and 2 can be added in several different ways. The minimum cost comes from adding 2+3 first to yield 5 and 5+6 to yield 11. The total cost for these two operations is 16 (5 + 11). Given a sequence of numbers calculate the minimum cost to add them into a single integer.

Input

First line of the input contains T the number of test cases. First line of each test case contains N the number of integers in the sequence. Second line contains N integers separated by a single space. N is between 1 and 20000 inclusive. Each of the integers in the sequence will be between 1 and 10000.

Output

For each test case output contains a single integer denoting the minimum cost.

Sample Input	Sample output
2 3 1 2 3 4 1 2 3 4	9 19

Conference Scheduling

Filename: *conference*
Time Limit: *5 seconds*

You are planning a large conference with many lectures. Lecturers submit their topic for the talk and in their submission they must specify the length of their talk. To be fair, as the conference organizer, you make sure that you schedule the lectures in a first come first served basis - you place the lectures in the order that the requests arrived, in the first room that is free.

You have an unlimited number of rooms you could use (so in theory, you could simultaneously place all lectures in different rooms), but each room costs a fixed amount of money so it's in your best interest to minimize the number of rooms you use for your schedule. Ideally, you could just have each of the lectures occur in one room, sequentially. But, the conference attendees are busy people and they can not necessarily afford to take weeks and weeks of time off. Instead, it is required that the conference finish in a fixed amount of time.

Thus, your goal is to schedule the conference so all lectures complete within a given fixed amount of time, minimizing the number of rooms used in the schedule.

Scheduling Details

Given that you are using k rooms for scheduling, number the rooms from 0 to k-1, inclusive. Here is how you should schedule each of the lectures:

Consider each lecture in the order the requests were received. Initially, all k rooms are free. Place the next lecture in the queue into the very first room that becomes free. If multiple rooms get free at precisely the same time, place the next lecture to be scheduled in the lowest numbered room of all of those available. Naturally, as a consequence of this algorithm, the first k lectures will be placed in rooms 0 through k-1, inclusive, each starting at time t = 0. This system will produce a single deterministic schedule, meaning that for each input case, once the correct value of k is determined, this algorithm will produce one correct answer.

Once you complete the scheduling, for each lecture you will have a room in which the lecture will be given and a starting time. This is the information you must output for each lecture. To earn full credit, you'll have to output each lecture in alphabetical order of lecture name.

Consider the following lecture requests, received in this order (lecture followed by time):

```
MATH 1000
BIOLOGY 2000
CHEMISTRY 1500
ENGLISH 1500
HISTORY 1300
PHYSICS 2200
SPANISH 1800
CIVICS 800
```

Let's say that we must complete the conference in 5000 seconds.

If we use three rooms, we get the following schedule:

Room 0 schedules MATH from time 0 to 1000.
Room 1 schedules BIOLOGY from time 0 to 2000.
Room 2 schedules CHEMISTRY from time 0 to 1500.
Room 0 schedules ENGLISH from time 1000 to 2500, since Room 0 frees up first.
Room 2 schedules HISTORY from time 1500 to 2800.
Room 1 schedules PHYSICS from time 2000 to time 4200.
Room 0 schedules SPANISH from time 2500 to 4300.
Room 2 schedules CIVICS from time 2800 to 3600.

To show that 3 is the correct minimum number of rooms, consider attempting to schedule the conference using 2 rooms:

Room 0 schedules MATH from time 0 to 1000.
Room 1 schedules BIOLOGY from time 0 to 2000.
Room 0 schedules CHEMISTRY from time 1000 to 2500.
Room 1 schedules ENGLISH from time 2000 to 3500.
Room 0 schedules HISTORY from time 2500 to 3800.
Room 1 schedules PHYSICS from time 3500 to 5700.
Room 0 schedules SPANISH from time 3800 to 5600.
Room 0 schedules CIVICS from time 5600 to 6400.

The Input

The first line of input will contain two space separated positive integers: n ($n \leq 10^5$), the number of lectures that must be scheduled for the conference, and t ($t \leq 10^9$) the maximum time allotted for the conference, in seconds. The information for each of the lectures follows, in the order that the lecture submissions were made. The next n lines contain the information about the lectures. Each of these lines contains two space separate pieces of information: the name of the lecture (a string of in between 1 and 19 uppercase letters) and the duration of that lecture (a positive integer less than or equal to 10^4.) in seconds. Each of the names of the events will be distinct so that the output order of the lectures is clearly specified.

The Output

The first line of output will be a single integer, k, the minimum number of rooms necessary to schedule all of the lectures so that the conference completes within the required time limit. This will be followed by n lines of output, one per lecture, in alphabetical order by lecture name. For each lecture, output the following pieces of information separated by spaces: the name of the lecture, the room that lecture is scheduled for, and the time (in seconds) after the beginning of the conference that the lecture will begin.

Sample Input

```
8 5000
MATH 1000
BIOLOGY 2000
CHEMISTRY 1500
ENGLISH 1500
HISTORY 1300
PHYSICS 2200
SPANISH 1800
CIVICS 800
```

Sample Output

```
3
BIOLOGY 1 0
CHEMISTRY 2 0
CIVICS 2 2800
ENGLISH 0 1000
HISTORY 2 1500
MATH 0 0
PHYSICS 1 2000
SPANISH 0 2500
```

Cow Dance

Filename: *cowdance*
Time Limit: *3 seconds*

After several months of rehearsal, the cows are just about ready to put on their annual dance performance; this year they are performing the famous bovine ballet "Cowpelia".

The only aspect of the show that remains to be determined is the size of the stage. A stage of size K can support K cows dancing simultaneously. The N cows in the herd ($1 \le N \le 10{,}000$) are conveniently numbered 1…N in the order in which they must appear in the dance. Each cow i plans to dance for a specific duration of time d(i). Initially, cows 1…K appear on stage and start dancing. When the first of these cows completes her part, she leaves the stage and cow K+1 immediately starts dancing, and so on, so there are always K cows dancing (until the end of the show, when we start to run out of cows). The show ends when the last cow completes her dancing part, at time T.

Clearly, the larger the value of K, the smaller the value of T. Since the show cannot last too long, you are given as input an upper bound T_{max} specifying the largest possible value of T. Subject to this constraint, please determine the smallest possible value of K.

The Input

The first line of input contains N and T_{max}, where T_{max} is an integer of value at most 1,000,000.

The next N lines give the durations d(1)…d(N) of the dancing parts for cows 1..N. Each d(i) value is an integer in the range 1..100,000.

It is guaranteed that if K = N, the show will finish in time.

The Output

Print out the smallest possible value of K such that the dance performance will take no more than T_{max} units of time.

SAMPLE INPUT:

```
5 8
4
7
8
6
4
```

SAMPLE OUTPUT:

```
4
```

Problem credits: Delphine and Brian Dean

G: Politics

You are helping a political party canvass their members. The party has a list of candidates they wish to support, and they also have a list of members, each of which supports one candidate.

The party wants for you to sort the list of members, according to some rules. You will be given the list of candidates, in the desired order. Then, you'll have access to a list of supporters; each supporter will have one candidate that they support.

The party wants you to put the list of supporters in order, sorted by their preferred candidate. There may be some supporters who support the same candidate. If so, order them in the order that they appear on the original list. Some supporters may have a 'write-in' candidate, thereby mentioning a candidate who was not on the original preferred candidate list. Put these supporters at the end, grouped by candidate, in the order that their *candidates* appear in the list of supporters. There also may be candidates who have no supporters. They will probably be taken off the list later, but that's just the politics of the situation.

Input

There will be several test cases in the input. Each test case will begin with a line with two integers, ***n*** ($1 \le n \le 1{,}000$) and ***m*** ($1 \le m \le 100{,}000$), where ***n*** is the number of candidates, and ***m*** is the number of supporters. On the following ***n*** lines will be the names of the candidates, one per line. These names will each be a single word consisting of from 1 to 30 capital letters. All ***n*** listed candidate names in any test case will be unique. After the candidates, the next ***m*** lines will describe the supporters, one per line. Each of these lines will contain two words separated with a single space. Each word contains from 1 to 30 capital letters. The first word is the name of the supporter, and the second is the name of the candidate that they support. The input will end with a line with two **0**s

Output

For each test case, print out the names of the supporters, one name per line, in the requested order. Do not output any spaces. Do not separate answers with a blank line.

Sample Input

```
3 5
STEVENS
MICHAELS
JORDAN
BOB JORDAN
JACK STEVENS
MACK MICHAELS
BILL JORDAN
CHRIS MATTHEWS
1 5
FRED
SAM FRED
ARTHUR GEORGE
DANIEL HERBERT
MARK GEORGE
MIKE HERBERT
0 0
```

Sample Output

```
JACK
MACK
BOB
BILL
CHRIS
SAM
ARTHUR
MARK
DANIEL
MIKE
```

Polling - More Votes!

Filename: *polling*
Time Limit: *2 seconds*

Midterm elections are here! Help your local election commission by counting votes and telling them the winner. If more than one candidate ties with the most votes, print out all of their names in alphabetical order.

The Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with an integer n ($1 \le n \le 10^5$), indicating the number of votes. The next n lines will hold the votes. The candidates' names will appear one per line, and consist of between 1 and 20 capital letters only.

The Output

Output the name of the candidate with the most votes. If there is a tie, output out all of the names of candidates with the most votes, one per line, in alphabetical order. Do not output any spaces, and do not output blank lines between names.

Sample Input 1

```
5
FRED
BARNEY
FRED
FRED
BARNEY
```

Sample Output 1

```
FRED
```

Sample Input 2

```
5
PORTHOS
ATHOS
ARAMIS
PORTHOS
ATHOS
```

Sample Output 2

```
ATHOS
PORTHOS
```

Problem C: Top 25

Filename: `top`

Timelimit: `8 seconds`

In College Football, many different sources create a list of the Top 25 (or, Top ***n***) teams in the country. Since it's subjective, these lists often differ, but they're usually very similar. Your job is to compare two of these lists, and determine where they are similar. In particular, you are to partition them into sets, where each set represents the same contiguous positions in both lists, and has the same teams, and is as small as possible. If the lists agree completely, you'll have ***n*** sets, where ***n*** is the number of teams in each list. For example consider these two lists:

For example consider these two lists:

A	A
B	C
C	D
D	B
E	E

In this case, there are 3 sets: A, BCD, and E.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs. Each test case will begin with an integer ***n*** ($1 \leq n \leq 1{,}000{,}000$), indicating the number of teams ranked. The next ***n*** lines will hold the first list, in order. The team names will appear one per line, and consist of between 1 and 8 capital letters only. After this will be ***n*** lines, in the same format, indicating the second list. Both lists will contain the same team names, and all ***n*** team names will be unique. .

Output

Output the size of each set, in order, one per line. Do not output any spaces, and do not output blank lines between numbers.

Samples

Input	Output
5 A B C D E A C D B E	1 3 1
3 RED BLUE ORANGE RED BLUE ORANGE	1 1 1
3 MOE LARRY CURLY CURLY MOE LARRY	3