Lund, November 18, 2017

Problems

Do not open before the contest has started.

Advice, hints, and general information

- Your submissions will be run multiple times, on several different input files. If your submission is incorrect, the error message you get will be the error exhibited on the first input file on which you failed. E.g., if your instance is prone to crash but also incorrect, your submission may be judged as either "wrong answer" or "run time error", depending on which is discovered first.

- For problems with floating point output, we only require that your output is correct up to some error tolerance.

  For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that

  - If the correct answer is 0.05, any answer between 0.0499 and $.0501$ will be accepted.
  - If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.

  Any reasonable format for floating point numbers is acceptable. For instance, "17.000000", "0.17e2", and "17" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Problem A
Arriving on Time
Problem ID: arrivingontime

We assume that this will never happen again in Zürich.
Photographer: Måns Magnusson

You are a very busy person, with a lot of important meetings. Today, you have a meeting for which it is insanely important to arrive at the agreed time.

Luckily you live in Zürich, which features a large network of extremely punctual trams. Each tram line travels from one place to another at regular intervals, always taking the same time from departure to arrival. It is very easy to change trams, and we assume that it takes no time to change to another tram if both are at a stop at the same time. This means that if a tram arrives at its destination at exactly time t and another tram departs from the same place at time t (or later), you will have enough time to change tram.

You are currently working in your hotel room before the meeting. Since you are a very busy person, you would like to leave your hotel at the latest possible time possible while still ariving to the meeting on time. When do you need to leave for your meeting?

Input

The input consists of:

- one line with three integers n, m and s ($2 \leq n \leq 100\,000$, $1 \leq m \leq 200\,000$, $1 \leq s \leq 10^9$), the number of tram stops, the number of tram lines, and the time at which the meeting starts in seconds relative to now.
- m lines, each with five integers u, v, t_0, p, d ($0 \leq u \neq v < n$, $0 \leq t_0 \leq 10^9$, $1 \leq p, d \leq 10^9$). The i'th line describes the i'th tram line, which departs from tram stop u, arrives at tram stop v, starts its first departure t_0 seconds from now, departs every p seconds from the first departure, and takes d seconds from departure to arrival.

The stops are numbered between 0 and $n-1$. Your hotel is located at stop 0, and the meeting is at stop $n-1$.

Output

Output the latest time at which you can leave the hotel while arriving to your meeting on time, in seconds from now. If you can not make it to your meeting on time, output `impossible` instead.

Sample Input 1	Sample Output 1
`2 1 10` `0 1 1 2 6`	`3`

Sample Input 2	Sample Output 2
`2 1 5` `0 1 1 1 5`	`impossible`

This page is intentionally left (almost) blank.

Problem B
Bus Numbers
Problem ID: busnumbers2

A famous story about the mathematicians *G.H. Hardy* and *Srinivasa Ramanujan* goes as follows (as told by Hardy):

> I remember once going to see him (Ramanujan) when he was lying ill at Putney. I had ridden in taxicab No. 1729, and remarked that the number seemed to be rather a dull one, and that I hoped it was not an unfavourable omen. "No", he replied, "it is a very interesting number; it is the smallest number expressible as the sum of two [positive] cubes in two different ways."

A bus with a number that is not a *bus number* according to our definition. License: public domain

It is from this story the *taxicab numbers* got their name. The n'th taxicab numbers is defined to be the smallest number that can be expressed as a sum of two *positive* cube numbers in n distinct ways.

It turns out that these numbers grows rather quickly. This makes them very hard to compute, which is not very fun. A variation of the concept is to consider what we will call the *bus numbers* – all the numbers which can expressed as the sum of two *positive* cube numbers in *at least* 2 distinct ways. Note that according to this definition, all taxicab numbers (except the first) are also bus numbers.

Your task is to write a program that generates bus numbers; in particular, the *largest* bus number that is *at most* equal to some limit m.

Input

The input consists of:

- one line with an integer m ($1 \leq m \leq 400\,000$), the upper bound of the bus number.

Output

Output the largest bus number x which does not exceed m. If there is no such number, output `none`.

Sample Input 1	Sample Output 1
`1730`	`1729`

Sample Input 2	Sample Output 2
`100`	`none`

This page is intentionally left (almost) blank.

Problem C
Cucumber Conundrum
Problem ID: cucumberconundrum

Maj loves pickled cucumber (also known as *pickles*). However, her partner is not as keen on filling a sandwich with pickles. Maj would like to maximize the amount of pickles on a sandwich, while still avoiding being judged by her partner.

Both Maj's sandwich and the pickles have a circular shape. The sandwich has radius s cm and the pickles have radius r cm.

Maj has exactly n pickles. She wants to place as many of them as possible on her sandwich, as long as:

- at most $z\%$ of the area of the sandwich is covered by pickles.
- no two pickles overlap (but they are allowed to touch).

How many pickles can Maj place on her sandwich?

A jar of sliced pickled cucumber. Author: W.carter, License: CC BY-SA 4.0

Input

The input consists of:

- one line with the decimal numbers s and r ($1 \leq s \leq 10$, $0.5 \leq r \leq s$, at most 6 digits after the decimal point), the radius of the sandwich and the radius of a pickle, in centimetres.
- one line with the integers n and z ($1 \leq n \leq 7$, $0 \leq z \leq 100$), the number of pickles Maj have, and the maximum area she may cover with them, in percent.

Output

Output the maximum number of pickles Maj can place on her sandwich. The input will always be constructed such that this number does not change if the radius of the sandwich increases or decreases by 10^{-6}.

Sample Input 1	Sample Output 1
3 1 4 40	3

Sample Input 2	Sample Output 2
3 1 4 100	4

This page is intentionally left (almost) blank.

Problem D
Distributing Seats
Problem ID: distributingseats

Divided airlines logo. Author: Måns Magnusson

An airline called *Divided Airlines* has recently made the news due to their tendency to overbook their flights rather aggressively. For some flights, they even resorted to dragging passengers out from the plane! This was of course not very popular, so they decided to "resolve" the issue by making the seating assignments very chaotic (airlines do like unnecessary complexity).

A particular flights has n passengers. The seats are divided into r rows each containing c seats. Every passenger i is assigned to some particular seat located at row a_i and column b_i. However, some passengers may be assigned to the same seat.

Of course, passengers are usually okay with sitting somewhere else than their assigned seat, but they may still want to be somewhat close to their original seat. Perhaps they want to be able to speak to their friends, or sit close to their overhead luggage. More specifically, passenger i accepts sitting at most s_i rows away from the row on their ticket.

Due to budget reasons, you decided to travel on a Divided flight. As expected, all the passengers assigned to an overbooked seat started to fight with each other, moving around in complex ways and causing a long delay. You proposed a fair resolution: you will construct a seat assignment which takes into account how far the passengers accepts to sit from their assigned seats so that as many passengers as possible get a seat. Now, all that remains is to actually find this assignment.

Input

The input consists of:

- one line with the integers n, r and c ($1 \leq n, r, c \leq 10^5$), the number of passengers, rows and columns in the flight.
- n lines with the integers a_i, b_i and s_i ($1 \leq a_i \leq r$, $1 \leq b_i \leq c$, $0 \leq s_i \leq r$). The i'th line has the assigned row a_i and column b_i, and maximum distance s_i of the i'th passenger. The maximum distance is given in rows.

Output

Output the maximum number of passengers that can be assigned a seat in an optimal assignment.

Sample Input 1	Sample Output 1
3 2 1 1 1 0 1 1 1 2 1 0	2

Sample Input 2

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

Sample Output 2

```
3
```

Sample Input 3

```
5 2 2
1 1 0
1 2 0
1 2 0
1 1 1
2 1 1
```

Sample Output 3

```
4
```

Problem E
Entering the Time
Problem ID: enteringthetime

A digital clock changing numbers. License: CC BY-SA 3.0. Author: Beyond silence.

Arghs! Yet again, all the clocks in Petra's home show the wrong time due to a power outage that occurred while she was sleeping. This means she has to spend her day resetting all the clocks to the correct time. Now, you might not consider this a big deal. After all, how many clocks does any single household really have? Well, Petra just so happens to be a collector of clocks. She literally has hundreds of clocks – and that is just in her bedroom! Thus, it is understandable that she does not look forward to all this resetting.

You, as her friend, would prefer if she did not need to spend her entire day resetting all her clocks. Especially since this means she would not have any time to play with you! If only you could construct some automated way of entering the correct time into all the clocks, perhaps through some kind of computer code...

A-ha, of course! You can write a program to determine how to set the correct time as quickly as possible!

Each clock has a display of 4 digits: two are used to display the hour (between `00` and `23`), and two are used to display the minute (between `00` and `59`). The time can be changed by selecting a digit and either decreasing or increasing by 1. Decreasing a digit that is 0, it is turned into 9, and increasing a digit 9 turns it into 0. However, the clock can not display invalid times. This means that at any given time, the hour must be between `00` and `23` and the minute between `00` and `59`.

Write a program that, given the original time of a clock and the current time, determines how to set the clock correctly.

Input

The input consists:

- one line with the time that the clock is currently set to.
- one line with the current time.

Each time has the format `hh:mm`, where `hh` is a two-digit number between `00` and `23`, and `mm` is a two-digit number between `00` and `59`.

Output

The first line contains the number of different times seen on the clock when setting it correctly. Then for each time output it on the same format as above `hh:mm` on a separate line. Include both the original time and the final time.

Sample Input 1	Sample Output 1

`00:00`
`01:01`
	`3`
	`00:00`
	`01:00`
	`01:01`

Sample Input 2	Sample Output 2
00:08 00:00	3 00:08 00:09 00:00

Sample Input 3	Sample Output 3
09:09 20:10	6 09:09 09:00 09:10 00:10 10:10 20:10

Problem F
Fishing Contest
Problem ID: fishingcontest

In a fishing contest, the participants fish in a lake, represented as a 2D grid of dimension $r \times c$. Each integer point in the grid contains fish.

At point (x, y), fish first appear at second $t_{x,y}$ and disappear just before time $t_{x,y} + k$ seconds. Outside of this time, no fish can be caught at this position. It takes no time to catch all the fish at a point, and all points contain the same amount of fish. Furthermore, moving to the point immediately north, west, south or east from the point you are currently at takes exactly 1 second.

Assume that you start at some position (x_0, y_0) at second 1, and can catch fish until (and including) second l. From how many points in the lake can you catch fish, if you travel optimally on the lake?

Ice fishing in Finland. License: CC BY-SA 3.0. Author: kallerna.

Input

The input consists of:

- one line with the integers r, c, k and l ($1 \leq r, c \leq 100$, $1 \leq k \leq 5$, $1 \leq l \leq 10^5$), the dimensions of the lake, the number of seconds fish stays at a point, and the number of seconds you can catch fish.
- one line with the integers x_0 and y_0 ($0 \leq r < x_0$, $0 \leq c < y_0$), your original position.
- r lines, the x'th of which contains c integers $t_{x,0}, \ldots, t_{x,c-1}$ (each between 1 and l, inclusive), the times at which fish appers on points in the x'th row.

Output

Output the maximum number of points you could catch fish from.

Sample Input 1

```
2 2 1 10
0 0
1 4
3 2
```

Sample Output 1

```
2
```

Sample Input 2

```
2 3 5 6
1 1
1 1 6
1 2 2
```

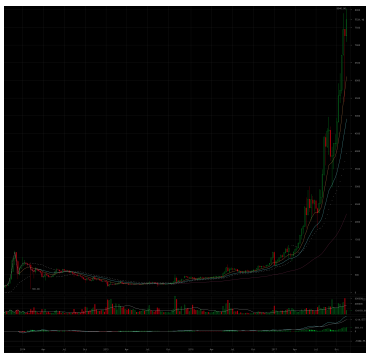
Sample Output 2

```
5
```

Sample Input 3	Sample Output 3
2 3 5 7 1 1 1 1 6 1 2 2	6

Problem G
Get-Rich-Quick Schemes
Problem ID: getrichquickschemes

After falling for a large number of fake get-rich-quick schemes, Marika is in serious need for cash, and really needs a get-rich-quick scheme. Instead of listenting to the ideas of strangers, Marika asked her mathematically talented friend David for some help. David suggested the following scheme, based on a feature of certain credit cards, called *cashback*.



Another way of getting rich quick could have been to invest in bitcoin prior to the summer of 2017. Author: Måns Magnusson.

Cashback means that you earn a certain percentage of cash back on purchase you make, depending on the type of purchase. Formally, there are n categories of merchandise. If you purchase merchandise from category i for x SEK, you earn $p_i \cdot x$ SEK back, up to some maximum limit m_i in a single month. While this does not help you earn money (since $p_i < 1$), you realized that you can simply return any products you bought to the store to get the money back.

Things are made more difficult by the fact that a particular store will get suspicious if you return too many products per month. In store i, you can buy and return merchandise for at most a_i SEK before start refusing you as a customer. Furthermore, a particular store only sells merchandise from a set of categories specific to that store. However, it has products costing any real amount of money from each category.

In order to get-rich-quick, Marika wants to earn as much money per month as possible. How much can she earn if she plans her purchases optimally?

Input

The input consists of:

- one line with the integer n ($1 \leq n \leq 300$), the number of categories.
- n lines with the integers p_i and m_i ($0 \leq p_i < 100$, $0 \leq m_i \leq 10^9$), the cashback rate (in percent) and the maximum limit of a product. The i'th line contains the rate and limit of the i'th product.
- one line with the integer s ($1 \leq s \leq 300$), the number of stores.
- s lines with the integers l_i and a_i, followed by a_i distinct integers $k_{i,1}, \ldots, k_{i,a_i}$, ($1 \leq l_i \leq 10^9$, $1 \leq a_i \leq n$, $1 \leq k_{i,j} \leq n$) The integers $k_{i,j}$ on the i'th line are the numbers of the categories sold by the i'th store. The categories are numbered between 1 and n in the order they are listed in the input.

Output

Output the maximum amount of money Marika can earn per month if purchasing items optimally. Your answer will be considered correct if it has a relative or absolute error of at most 10^{-6}.

Sample Input 1	Sample Output 1
3 10 100 20 50 15 40 5 20 3 1 2 3 20 2 2 3 20 1 2 20 1 3 20 2 1 2	17

Problem H
Hide and Seek
Problem ID: hideandseek

Playing hide and seek in minecraft. Author: icebingo

Sven the Polar Bear (the mascot of the various Swedish national teams in science sports) plays hide and seek with Gloria the Hippo (the mascot from Madagascar). The game is played in a network of m caves, numbered between 0 and $m-1$. The caves are connected by $m-1$ tunnels (each between two distinct caves), such that there is a path between any pair of caves.

Each round starts with Gloria hiding in one of the caves (except cave 0) with uniform probability. Sven, starting in cave 0, then has n seconds to find Gloria. Each tunnel takes some number of seconds for Sven to traverse depending on the length of the tunnel, and can be traversed in both directions.

Sven likes to win, so he wants to choose his movements in such a way that he maximizes the probability of finding Gloria before the n seconds are up. How many caves does Sven have time to visit?

Input

The input consists of:

- one line with the integers m and n ($2 \leq m \leq 100$, $1 \leq n \leq 300$), the number of caves in the network and the number of seconds Sven has to find Gloria.
- $m-1$ lines with three integers u, v and t ($0 \leq u \neq v < m$, $1 \leq t \leq 300$), the two endpoints of a tunnel and the time it takes to traverse the tunnel, in seconds.

Output

Output the maximum number of caves Sven can visit, excluding the cave he starts in.

Sample Input 1	Sample Output 1
11 50 0 1 5 1 2 4 1 3 9 0 4 6 4 5 6 5 6 3 0 7 8 7 8 10 7 9 5 9 10 2	6

This page is intentionally left (almost) blank.

Problem I
#include<scoring>
Problem ID: includescoring

The logotype of the Swedish Coding Cup.

As you may know, LTH Challenge is part of a series of seven competitions called the *Swedish Coding Cup*. Each contest in the series gives the contestants a number of points depending on how well they place in the contest.

After each contest, scores are assigned according to the following table:

Rank	Score	Rank	Score
1	100	16	15
2	75	17	14
3	60	18	13
4	50	19	12
5	45	20	11
6	40	21	10
7	36	22	9
8	32	23	8
9	29	24	7
10	26	25	6
11	24	26	5
12	22	27	4
13	20	28	3
14	18	29	2
15	16	30	1

If a contestant get a worse rank than 30, they get 0 points.

If two or more contestants get the same rank in the contest, they are instead assigned the average score of all the corresponding ranks. This score is always rounded up to the closest integer. For example, if three contestants share the second place they all recieve $\lceil \frac{75+60+50}{3} \rceil = 62$ points.

Contestants may participate in every contest either on-site or online. If they compete on-site, they get one extra point, no matter their original score. If a contestant does not participate in a contest, they are assigned a score of 0.

At LTH Challenge, the rank of each contestant is computed using what is called *ACM scoring*. Each contestant is ranked first on the number of problems they solve (in descending order), secondly on their time penalty (in ascending order) and finally the time at which they submitted their last accepted solution in minutes (in ascending order). If all these three properties are equal, the contestants are tied.

As you may understand, it is really tedious to compute the scores of all the contestants after such a contest. The jury does not really want to program this themselves; they prefer to just use some existing library instead. However, it turns out that since this is the first time the Swedish Coding Cup is held, nobody had written such a library!

Fortunately, they have you.

Input

The input consists of:

- one line with the integer n ($1 \leq n \leq 1\,000$), the number of contestants.
- n lines with the integers s, p, f and o, ($0 \leq s \leq 9$, $0 \leq p \leq 10^9$, $0 \leq f \leq 300$, $0 \leq o \leq 1$) – the number of problems solved by a contestant, the time penalty of the contestant, the time at which they submitted their last accepted solution and the number of extra points the contestant should get due to competing on-site.

Output

Output n lines containing the scores of all the contestants in the order they were listed in the input.

Sample Input 1	Sample Output 1
41 6 179 65 1 6 305 86 1 6 324 96 0 6 390 112 1 5 280 97 0 4 79 45 1 4 94 49 1 4 126 55 1 4 160 100 1 4 173 76 0 4 214 106 1 4 221 110 1 4 226 96 1 4 384 103 0 3 35 26 1 3 90 56 0 3 113 83 1 3 137 106 0 3 171 101 1 3 184 104 0 3 195 65 1 2 14 11 1 2 15 10 1 2 17 11 1 2 19 12 1 2 20 13 1 2 21 14 0 2 29 20 1 2 30 19 0 2 36 27 1 2 39 13 1 2 52 23 1 2 52 31 1 2 69 23 1 2 86 17 0 2 113 55 1 2 125 62 1 2 328 119 0 1 10 10 0 1 15 15 1 1 33 33 0	101 76 60 51 45 41 37 33 30 26 25 23 21 18 17 15 15 13 13 11 11 10 9 8 7 6 4 4 2 2 1 1 1 1 0 1 1 0 0 1 0

This page is intentionally left (almost) blank.