Practice Contest for ACM-ICPC SouthWestern Europe Regional Contest 2017

Paris, 25 November 2017

acm International Collegiate Programming Contest

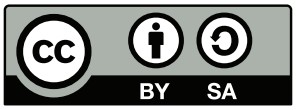
SOCIETE GENERALE
Palantir
TELECOM ParisTech
ENS
criteo labs
Google
almerys
Inover pour la vie
Inria

Judges and Problem Setters

- Mehdi Bouaziz (Facebook)
- Christoph Dürr (CNRS, LIP6)
- Jean-Christophe Filliâtre (CNRS, LRI)
- Thomas Huet (Google)
- Ioana Ileana (Université Paris-Descartes)
- Louis Jachiet (École normale supérieure)
- Jacques-Henri Jourdan (CNRS, LRI)
- Silviu Maniu (Université Paris-Sud)
- Raphaël Marinier (Google)
- Pierre Senellart (École normale supérieure), *chief judge*
- Samuel Tardieu (Télécom ParisTech)

This problem set consists of 5 problems, on 11 pages.

This work is licensed under a Creative Commons "Attribution-ShareAlike 4.0 International" license.



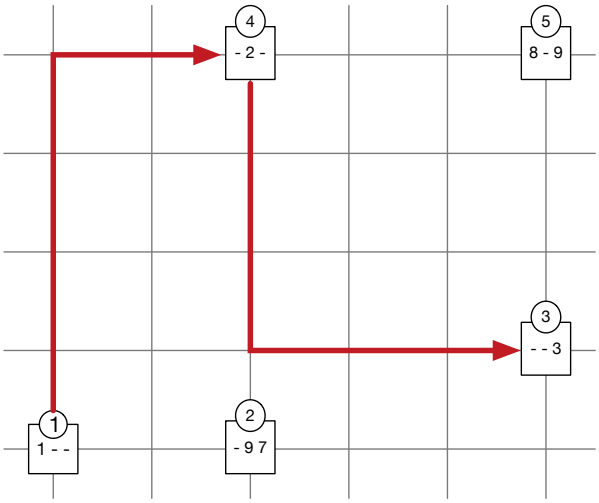
A: Menu Tour

Rachid is in Barcelona for the first time, and wants to have a really good dinner. He heard that the ultimate dinner consists of C courses, which are numbered from 1 to C and have to be consumed in that order.

The streets of Barcelona form a regular grid: west–east oriented streets crossing south–north oriented streets. There are R restaurants in the city and they are located at the crossings. Walking from crossing (i_1, j_1) to crossing (i_2, j_2) takes exactly $|i_1 - i_2| + |j_1 - j_2|$ minutes, where $|x|$ denotes the absolute value of x. Here, (i, j) means ith street from west to east, and jth street from south to north.

Unfortunately, restaurants do not offer all of the C courses. If a restaurant k offers course c then it costs $P[k, c]$ euros. Otherwise the value $P[k, c] = 0$ indicates that the course is not offered. Rachid has B euros that he can spend on his dinner. He would like to choose a sequence of restaurants so that he can have his ultimate dinner without exceeding the available budget, while minimizing the total travel time between restaurants. The tour can start and end at an arbitrary crossing, and can visit the same restaurant several times.

Example



On this example, there are three courses, whose prices are displayed in order, for every restaurant (a "-" corresponds to a dish that is not offered, i.e., when $P[k, c] = 0$). If Rachid has a budget of 9 euros, the optimal tour consists of the restaurants 1–4–3 for a cost of 6 euros and a total travel time of 12 minutes. There is a shorter tour of 2 minutes consisting of the restaurant sequence 1–2–2, but its cost is 17 euros, exceeding the available budget of 9 euros.

Input

The input begins with a line consisting of the integers C, R, B, separated by a single space. Then R lines follow. The k-th line describes the k-th restaurant and consists of $2 + C$ integers separated by a single space, namely $i[k], j[k], P[k, 1], \ldots, P[k, C]$, where $(i[k], j[k])$ defines the location of the restaurant.

Limits

- $1 \leqslant C \leqslant 20$;
- $1 \leqslant R \leqslant 100$;
- $0 \leqslant B \leqslant 100$;

- for every $1 \leqslant k \leqslant R$:
  - $1 \leqslant i[k] \leqslant 1\,000$;
  - $1 \leqslant j[k] \leqslant 1\,000$;
  - for every $1 \leqslant c \leqslant C$, $0 \leqslant P[k, c] \leqslant 40$.

Output

The output should consist of a single integer y: the minimum total travel time of an optimal restaurant tour for Rachid. If there is no feasible tour, the value `-1` should be printed.

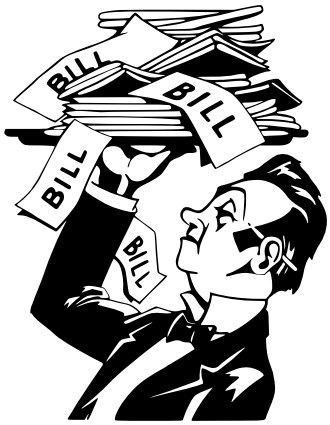
Sample Input

```
3 5 9
1 1 1 0 0
3 1 0 9 7
6 2 0 0 3
3 5 0 2 0
6 5 8 0 9
```

Sample Output

```
12
```

B: Check the Check



As a tourist in Paris, you were told you should always carefully check the itemized bill (also called *check*) that is presented to you at the end of a meal with the list of what you ordered and the total price. Indeed, it is not uncommon for these bills to be handwritten, and for the total to be computed by hand by the waiter. You definitely do not want to overpay for your meal, and will protest if there is a mistake in the restaurant's favor. However, if the restaurant gives you a discount, you will not complain about it.

Write a program that decides whether you should pay the total amount presented on the check, or protest about the check.

Input

The input is formed of $2n + 2$ lines:

- lines $2k + 1$ for $0 \leqslant k \leqslant n - 1$ consist of the name of the ordered dish d_k;
- lines $2k + 2$ for $0 \leqslant k \leqslant n - 1$ consist of the integer price p_k of d_k in euros, and the number c_k of orders for d_k, separated by a space;
- line $2n + 1$ consists of the word "TOTAL";
- line $2n + 2$ consists of the integer total T in euros computed by the waiter.

Limits

- For every $0 \leqslant k \leqslant n - 1$:
  - d_k has at most 1 000 characters, and is never equal to "TOTAL";
  - $0 \leqslant p_k \leqslant 1\,000$;
  - $0 \leqslant c_k \leqslant 10$;
- $0 \leqslant n \leqslant 100\,000$;
- $T \leqslant 2\,000\,000\,000$.

Output

The output should consist of a single line, whose content is either "PAY" (if the displayed total is less than or equal to the actual total) or "PROTEST" (otherwise).

Sample Input

```
Foie gras
15 2
Huîtres
10 1
Bœuf bourguignon
18 1
Magret de canard
17 1
Lapin à la moutarde
16 1
Crème brûlée
6 1
Mousse au chocolat
5 2
TOTAL
100
```

Sample Output

```
PAY
```

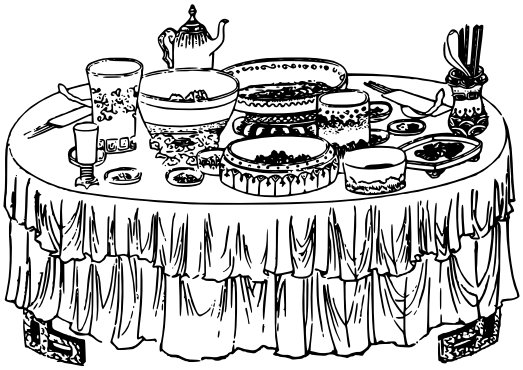
Sample Input

```
Escargots de Bourgogne
15 2
Pâté en croûte
10 1
Blanquette de veau
18 1
Gratin dauphinois
17 1
Ratatouille
16 1
Profiteroles
6 1
Crêpe au sucre
5 2
TOTAL
108
```

Sample Output

```
PROTEST
```

C: How to Eat at a Buffet



Instead of a seated meal, some restaurants offer all-you-can-eat buffet lunches and dinners. This is usually a bargain for famished students. Alice likes such buffet meals, but is always worried about how best to fill her plate with food. She values the n different items on the menu differently, and her goal is to have as much value as she can on her plate, constrained by the limited area of the plate and the limited availability of some menu items. Luckily, items on the menu are easily dividable and Alice can take an arbitrary fraction of each dish. Can you help her fill her plate?

Input

The input is formed of $n + 2$ lines:

- the first line consists of the number n of different items on the menu;
- the second line consists of the area a of Alice's plate, an integer in mm^2;
- each of the n remaining lines consists of information about a menu item, as two integers separated with a space; the first one is the value v_i of item i per mm^2, as perceived by Alice; the second one is the area a_i, in mm^2, that item i would occupy if Alice were to transfer it fully to her plate.

Limits

- $1 \leqslant n \leqslant 1\,000$
- $0 \leqslant a \leqslant 100\,000$
- For every $0 \leqslant i \leqslant n - 1$:
  - $0 \leqslant v_i \leqslant 100$;
  - $0 \leqslant a_i \leqslant 100\,000\,000$.

Output

A single line consisting of an integer: the maximal value that Alice can fit on her plate.

Sample Input

```
5
1000
50 230
80 12
10 1000000
25 450
2 50
```

Sample Output

```
26790
```

D: French Dinner

You are a French chef and need to plan a very prestigious meal. As a French chef, you know all these weird politeness rules the French have on meals: which items must be served simultaneously, which need to be served first, how long you must wait between courses, etc.

For instance, the red wine accompanying the cheese must be served at least five minutes before the cheese plate arrives, otherwise people cannot choose what they consider the most adequate cheese to go with the wine. In addition, this wine needs to arrive at least twenty minutes after the main course, so people have time to finish the wine that has been served with the main course. And the cheese needs to be served around the same time as bread, otherwise people will eat bread and cheese separately… and so many other rules that you are not even sure you can satisfy all of them.

Fortunately, you have written all those rules down and you are now trying to organize the perfect meal. There are two kinds of rules:

- rules of simultaneity (`SIM` *A B T*) indicating that *A* and *B* must be served at most *T* minutes apart;
- rules of precedence (`BEF` *A B T*) indicating that *A* must be served at least *T* minutes before *B*.

You also know that a meal should always take at most *K* minutes in total (there must be at most *K* minutes between the first and last items served) and no dishes should ever be skipped.

Note that the constraints given are always inclusive of the bounds. For example, when *A* is served at $t = 0$ and *B* is served at $t = 1$, then both `SIM` *A B* `1` and `BEF` *A B* `1` are satified, and the meal composed of *A* and *B* takes 1 minute.

Your job is to help the chef determine whether it is possible to plan a meal that satisfies all rules.

Input

The input is formed of $N + 1$ lines:

- the first line consists of two integers *N* and *K*, separated with a space. *N* is the number of rules and *K* is the maximal duration of the meal;
- each of the *N* remaining lines consists of a rule either of the form `SIM` *A B T* or of the form `BEF` *A B T* where *T* is an integer and *A* and *B* are non-empty strings without spaces of at most 1 000 characters.

Limits

All given integers (N, K, the T's) are between 0 and 1 000 inclusive.

Output

The output should consist of a single line, whose content is either "YES" (if it is possible to organize a meal following all given constraints) or "NO" (otherwise).

Sample Input

```
7 150
BEF Hors_d_oeuvre Appetizer 30
BEF Appetizer Main_course 30
SIM Main_course Bread_1 5
BEF Main_course Cheese 30
SIM Cheese Bread_2 5
BEF Cheese Dessert 30
BEF Bread_1 Bread_2 45
```

Sample Output

```
YES
```

Sample Input

```
7 120
BEF Hors_d_oeuvre Appetizer 30
BEF Appetizer Main_course 30
SIM Main_course Bread_1 5
BEF Main_course Cheese 30
SIM Cheese Bread_2 5
BEF Cheese Dessert 30
BEF Bread_1 Bread_2 45
```

Sample Output

```
NO
```

E: Majestic Gourmet University

The Majestic Gourmet University offers the prestigious Majestic International Cooking Degree to a large number of students each year.

Every student has to take two mandatory classes: French Cuisine (FC) and Italian Cuisine (IC), taught by two disjoint groups of teachers: FC teachers and IC teachers. These classes take the form of weekly "cooking labs", and have limited capacity: K^{FC} students for FC labs and K^{IC} students for IC labs. To accommodate all students, there are therefore several lab instances for both FC and IC every week, each with its specific weekly timeslot and assigned teacher. One added complication is that some of the teachers are in open conflict with each other, disagreeing on teaching methods.

Every year, the heads of the FC and IC departments come up with a number of propositions for the weekly lab instances, and let Wise Bob from the Planning Office choose a subset of these propositions so that each student can have a valid timetable, which means:

- Every student must attend exactly one weekly FC lab and one weekly IC lab. Note that a student cannot attend two labs if their timeslots overlap or if they are separated by less than 5 minutes, to leave time for switching classrooms.
- Maximal capacities must be respected (at most K^{FC} students in an FC lab, and at most K^{IC} students in an IC lab).
- If an FC teacher A has a conflict with an IC teacher B, no student of a lab taught by A can attend a lab taught by B.

Besides selecting weekly lab instances to ensure a proper lab assignment for all students, Wise Bob further wishes to ensure a maximum number of "free days" during the week. Accordingly, an optimal solution for Bob is one such that, regardless of the number of lab instances proposed, the starting time of their corresponding slots are spread over a minimum number of (not necessarily consecutive) days of the week.

Wise Bob asks for your help to figure out which lab slots to pick and how to dispatch students to these labs, so as to achieve an optimal solution.

Note that some teachers may not be appointed to any lab instances, while others may be in charge of several labs.

Input

The input comprises the following lines all consisting of integers separated by single spaces:

- **Line 1** consists of S, the number of registered students.
- **Line 2** consists of N^{FC}, K^{FC}, D^{FC} and T^{FC}, where N^{FC} is the number of FC lab propositions, K^{FC} is the maximal capacity of each FC lab, D^{FC} is the duration (in hours) of each FC lab, and T^{FC} is the number of FC teachers.
- **Each line from line 3 to line $N^{\text{FC}} + 2$** consists of the description of a proposed FC lab slot. FC slot i is described by four integers D_i^{FC}, H_i^{FC}, M_i^{FC}, T_i^{FC}, where D_i^{FC} is the day of the week, $(H_i^{\text{FC}}, M_i^{\text{FC}})$ is the starting time of the lab (hour, minutes) and T_i^{FC} is the FC teacher appointed.
- **Line $N^{\text{FC}} + 3$** consists of N^{IC}, K^{IC}, D^{IC} and T^{IC}, where N^{IC} is the number of IC lab propositions, K^{IC} is the maximal capacity of each IC lab, D^{IC} is the duration (in hours) of each IC lab, and T^{IC} is the number of IC teachers.
- **Each line from line $N^{\text{FC}} + 4$ to line $N^{\text{FC}} + N^{\text{IC}} + 3$** consists of the description of a proposed IC lab slot. IC slot i is described by four integers D_i^{IC}, H_i^{IC}, M_i^{IC}, T_i^{IC}, with the same meanings as for the FC slots.
- **Line $N^{\text{FC}} + N^{\text{IC}} + 4$** consists of the number C of conflicts among teachers.
- **Each of the last C lines** consists of two integers i and j ($0 \leqslant i < T^{\text{FC}}$, $0 \leqslant j < T^{\text{IC}}$), meaning that FC teacher i has a conflict with IC teacher j.

Output

The output consists of an integer on a single line. This integer is either (i) 0, if no solution exists, or (ii) the minimal number of days of the week that contain the starting time of slots in an optimal solution.

Limits

- $1 \leqslant S \leqslant 11\,000$ and $1 \leqslant K^{\mathrm{FC}}, K^{\mathrm{IC}} \leqslant 64$;
- $1 \leqslant N^{\mathrm{FC}}, N^{\mathrm{IC}}, T^{\mathrm{FC}}, T^{\mathrm{IC}} \leqslant 1\,000$;
- $1 \leqslant D^{\mathrm{FC}}, D^{\mathrm{IC}} \leqslant 8$;
- $0 \leqslant C \leqslant T^{\mathrm{FC}} \times T^{\mathrm{IC}}$;
- $1 \leqslant D_i^{\mathrm{FC}}, D_i^{\mathrm{IC}} \leqslant 6$;
- $8 \leqslant H_i^{\mathrm{FC}}, H_i^{\mathrm{IC}} \leqslant 20$ and $0 \leqslant M_i^{\mathrm{FC}}, M_i^{\mathrm{IC}} \leqslant 59$;
- $0 \leqslant T_i^{\mathrm{FC}} < T^{\mathrm{FC}}$ and $0 \leqslant T_i^{\mathrm{IC}} < T^{\mathrm{IC}}$.

Note that no two propositions for the same teacher may overlap.

Sample Input

```
90
2 45 2 2
1 9 0 0
2 15 0 1
5 30 2 2
1 8 0 0
2 14 0 0
3 15 15 0
1 16 25 0
1 17 10 1
1
0 1
```

Sample Output

```
2
```

Sample Input

```
50
2 30 2 2
1 9 0 0
2 10 10 1
2 40 2 2
1 10 40 0
2 12 10 1
1
1 0
```

Sample Output

```
0
```