Speed Limits

In our busy world, we are not usually interested in taking the shortest path, but rather the path that takes the shortest time. When driving a car, this means that the speed limits on different roads are crucial. Imagine now that some speed limit signs are missing. Since you cannot expect the driver to know speed limits by heart, the only reasonable conclusion is that whatever speed limit he/she obeyed before will still hold after passing a missing sign. You are to write a program that calculates the fastest path by taking advantage of the missing signs.

You are given a description of the road network in a highly motorized area. To make things easier the network consists of *crossings* and *roads*. Every road is one-way, connects exactly two crossings and has at most one speed limit sign, located in the beginning of the road. For any two crossings A and B, there exists at most one road from A to B. You may assume that acceleration is instantaneous and that no other traffic will affect you. And of course you never drive faster than the current speed limit.

Input data

The first line of the input file `speed.in` consists of three integers N, M and D, where N ($2 \le N \le 150$) is the number of crossings numbered from 0 to N-1, M is the number of roads and D is the label of the crossing that is your destination. Each of the following M lines of the input file describes one road. Every line consists of four integers A ($0 \le A < N$), B ($0 \le B < N$), V ($0 \le V \le 500$) and L ($1 \le L \le 500$), signifying that the road goes from the crossing labelled by A to crossing B, has speed limit V and length L. If V is zero, this means that the speed limit sign is missing. The time T taken for a given road is thus $T = L / V$ if $V \neq 0$, otherwise $T = L / V_{old}$, where V_{old} is the speed limit you obeyed before you arrived at the crossing. Note that the division should be done with floating-point numbers to avoid unnecessary rounding. In the beginning you are at crossing 0 and your current speed is 70.

Output data

The output file `speed.out` must consist of one single line of integers, describing the crossings you pass on the fastest possible path from 0 to D. The crossings should be written in the exact order in which you pass them, starting with 0 and ending with D. There will never be two fastest paths taking the same time.

Example

speed.in

```
6 15 1
0 1 25 68
0 2 30 50
0 5 0 101
1 2 70 77
1 3 35 42
2 0 0 22
2 1 40 86
2 3 0 23
2 4 45 40
3 1 64 14
3 5 0 23
4 1 95 8
5 1 0 84
5 2 90 64
5 3 36 40
```

speed.out

```
0 5 2 3 1
```

Comments

The required time is in this case 2,628 units.

Tennis Club

The Matchball tennis club is organizing a "game interest week" to attract new players to the club. As one of the attractions, they have asked some star players to play a few demo games. Each star has indicated the number of games he or she is willing to play. The organizers want the stars to have some fun as well, thus they want to schedule the games so that no two players meet more than once with each other.

Your task is to write a program to help them match the players into pairs so that each player plays his or her desired number of games and does not play twice or more against any other player. Of course, no player may play against himself or herself.

Input data

On the first line of the input file `tennis.in` is the number of players N ($2 \le N \le 1000$) and on the following N lines is the desired number of games to play G_i ($1 \le G_i < N$) for each player. Assume that the players are numbered from 1 to N in the order of their wishes in the input file.

Output data

On the first line of the output file `tennis.out` write `NO SCHEDULE` if it is not possible to create a schedule so that the wishes of all players are satisfied, or `SCHEDULE` if it is possible.

If a schedule exists, write it out on the following N lines. On each line write the indices of opponents for the player whose desired number of games was indicated on the corresponding input line. On each line the indices must be in increasing order and separated by spaces. If multiple solutions exist, output any one of them.

Examples

tennis.in	tennis.out
3 1 2 1	SCHEDULE 2 1 3 2

tennis.in	tennis.out
3 2 2 1	NO SCHEDULE

Grading

In this task, a program receives points for tests with no schedule only if it solves correctly at least half of the tests with a schedule.

Triangles

There are given n isosceles right triangles on a plane. Each triangle can be described by three integers x,y,m (m>0). Vertices of such a triangle are points which have coordinates (x;y), (x+m;y) and (x; y+m).

Write a program which calculates the total area covered by triangles.

Input data

The first line of the input file `tr.in` contains one positive integer n ($n \le 2000$), the number of triangles on a plane .

The next n lines of the file describe the triangles, one triangle per line. Each line contains three integers x_i, y_i and m_i, separated by single spaces ($1 \le i \le n$, $-10^7 \le x_i \le 10^7$, $-10^7 \le y_i \le 10^7$, $0 < m_i \le 1000$).

Output data

On the first line of the file `tr.out` one number with exactly one digit after decimal point must be written – the total area covered by triangles.

Example

tr.in

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

tr.out

```
24.5
```