Day 2 Question 1: Duathlon

Your solution: N:\duathlon\ duathlon.{pas, c, cpp}
Input file: duathlon.in
Output file: duathlon.out

A duathlon is a race that involves running *r* km and cycling *k* km. *n* contestants have entered the race; each contestant has different running and cycling speeds. One of the contestants has bribed the organizers to set *r* and *k* so that he can win by the maximum margin. You are to determine if this is possible and, if so, give *r* and *k*.

The first line of input contains an integer *t*, the total distance of the race, in km. That is, $r + k = t$. The next line contains an integer *n*, the number of competitors. For each contestant, a line follows with two real numbers giving the running and cycling speed for that contestant. The last line of input gives the running and cycling speed of the contestant who has bribed the organizers. You may assume *t* does not exceed 100 km/h and *n* does not exceed 20.

If it is possible to fix the race as describe above, print a message giving *r* and *k*, and the amount by which the cheater will win the race, accurate to two decimal places as in the sample below. If it is not possible, print "`The cheater cannot win.`"

Sample Input 1

```
100
3
10.0 40.0
20.0 30.0
15.0 35.0
```

Output for Sample Input 1

```
The cheater can win by 612 seconds with r = 14.29km and k = 85.71km.
```

Sample Input 2

```
100
3
10.0 40.0
20.0 30.0
15.0 25.0
```

Output for Sample Input 2

```
The cheater cannot win.
```

Day 2 Question 2: Connect the Campus

Your solution: N:\connect\connect.{pas, c, cpp}
Input file: connect.in
Output file: connect.out

Many new buildings are under construction on the campus of the University of Polkaroo. The university has hired bricklayers, electricians, plumbers, and a computer programmer. A computer programmer? Yes, you have been hired to ensure that each building is connected to every other building (directly or indirectly) through the campus network of communication cables.

We will treat each building as a point specified by an x-coordinate and a y-coordinate. Each communication cable connects exactly two buildings, following a straight line between the buildings. Information travels along a cable in both directions. Cables can freely cross each other, but they are only connected together at their endpoints (at buildings).

You have been given a campus map which shows the locations of all buildings and existing communication cables. You may not alter the existing cables. Determine where to install new communication cables so that all buildings are connected. Of course, the university wants you to minimize the amount of new cable that you use.

The input file describes one test case. The first line contains the number of buildings N (1 <= N <= 750). The buildings are labelled from 1 to N. The next N lines give the x and y coordinates of the buildings. These coordinates are integers with absolute values at most 10000. No two buildings occupy the same point.

Finally there is a line containing the number of existing cables M (0 <= M <= 1000) followed by M lines describing the existing cables. Each cable is represented by two integers: the building numbers which are directly connected by the cable. There is at most one cable directly connecting each pair of buildings.

For each new cable you plan to construct, output a line containing the numbers of the two buildings that will be joined by the cable. Also output the total length of the new cables that you plan to use, rounded to two decimal places. If there is more than one possible minimal solution you may output any such solution.

Sample Input

```
4
103 104
104 100
104 103
100 100
1
4 2
```

Output for Sample Input

```
4.41
3 2
1 3
```

Day 2 Question 3: Pit Stop Strategy

Your solution: N:\pitstop\pitstop.{pas, c, cpp}
Input file: pitstop.in
Output file: pitstop.out

The speed of a racing car, all other factors being equal, depends on the amount of fuel it carries. In general, the weight of the fuel slows the car. In addition, the weight of the fuel increases fuel consumption. It is therefore an advantage to carry as little fuel as possible. Any amount of fuel may be loaded at the beginning of the race, and refueling pit stops may be made during the race. The time consumed for the pit stop increases with the amount of fuel loaded. Your task is to determine the optimal fueling and pit stop strategy for each of a number of cars, based on measurements taken immediately before the race.

Input

Standard input consists of several lines of input, each containing:

- the number of laps in the race (integer less than or equal to 100)
- the theoretical lap time [seconds] of the car on an empty tank (float)
- the increase in lap time [seconds] per litre of fuel carried at the beginning of the lap (float)
- the theoretical fuel consumption [litres per lap] on an empty tank (float)
- the increase in fuel consumption per litre of fuel carried at the beginning of the lap (float; strictly less than 1)
- the time [seconds] for a pit stop taking on no fuel (float)
- the extra pit stop time per litre of fuel loaded (float)

Output

For each input line, print the following information

- one line containing the seven input numbers in order
- one line containing three integers:
    1. total race time
    2. the amount of fuel to be loaded initially
    3. the number of pit stops
- for each pit stop, a line containing:
    1. the number of laps completed since the start of the race at the time of the pit stop
    2. the amount of fuel to be taken on

All floating point results should be printed to 3 decimal places. All numbers on a single line should be separated by a single space.

Sample Input

```
3 100 0 10 0 20 0
3 100 0 10 .1 20 0
3 100 2 10 0 20 1
```

```
3 100 4 10 0 20 1
3 100 2 10 .1 20 1
```

Output for Sample Input

```
3 100.000 0.000 10.000 0.000 20.000 0.000
300.000 30.000 0
3 100.000 0.000 10.000 0.100 20.000 0.000
300.000 37.174 0
3 100.000 2.000 10.000 0.000 20.000 1.000
410.000 20.000 1
2 10.000
3 100.000 4.000 10.000 0.000 20.000 1.000
480.000 10.000 2
1 10.000
2 10.000
3 100.000 2.000 10.000 0.1000 20.000 1.000
422.469 23.457 1
2 11.111
```