The 25th Korea Olympiad in Informatics (KOI)

2008

<Elementary School Competition>

Trimmed Mean

In sports such as gymnastics or diving, sometimes a judge tends to give biased scores depending on his likes or dislikes on an athlete. By including these biased scores to an averaged score, it is unfair to the player. To prevent this situation, trimmed mean or Winsorized mean is used.

For example, seven judges gave the following score:

9.3, 9.5, 9.6, 9.8, 9.1, 5.0, 9.3

The total sum is 61.6, which gives the average to be 8.8. This average is a result of one low score of 5.0 that may seem unfair to the contestant.
The above scores are sorted in ascending order:

5.0, 9.1, 9.3, 9.3, 9.5, 9.6, 9.8

Trimmed_Mean(7,2) represents discarding two lowest and two highest scores and calculating the average only without those, 9.3, 9.3, and 9.5, which gives the average of 9.37 rounding up to two decimal places.
Winsorized_Mean(7,2) represents, in ascending order, replacing two lowest and two highest score with the next closest score: 9.3, 9.3, 9.3, 9.3, 9.5, 9.5, 9.5. Now the average becomes 9.39 rounding up to two decimal places.

Write a program that calculates **Trimmed_Mean(N,K)** and **Winsorized_Mean(N,K)** where N represents the total number of scores and K represents number of scores getting discarded or replaced from each side in ascending order.

The maximum running time is 0.5 seconds.

Input Format (Input.txt)
The first line has the values of N ($3 \le N \le 100{,}000$) and K ($0 \le K \le (N/2)-1$) with a space in between. Then there are N lines with each score on each line. The scores are real numbers ($0 \le score \le 10$) rounds up to one decimal place.

Output Format (Output.txt)
The first line should print out the result from **Trimmed_Mean(N,K)** and second line from **Winsorized_Mean(N,K)** with the answers round up to two decimal places. For example, if the answer is 9.667, round up to 9.67, 5 to 5.00, and 5.5 to 5.50.

Example Input and Output

INPUT.TXT

```
7 2
9.3
9.5
9.6
9.8
9.1
5.0
9.3
```

OUTPUT.TXT

```
9.37
9.39
```

World Cup

In World Cup final qualification tournament, six countries make a group to match against the countries in a same group so that each country has total of five games. After the tournament, a reporter sent the results of the games. We are to determine if the number of wins, draws, and loses of the countries are possible.

The followings are examples of possible and impossible results.

<Example 1> Possible Result

Country	Win	Draw	Lose
A	5	0	0
B	3	0	2
C	2	0	3
D	0	0	5
E	4	0	1
F	1	0	4

<Example 2> Possible Result

Country	Win	Draw	Lose
A	4	1	0
B	3	0	2
C	4	1	0
D	1	1	3
E	0	0	5
F	1	1	3

<Example 3> Impossible Result

Country	Win	Draw	Lose
A	5	0	0
B	4	0	1
C	2	2	1
D	2	0	3
E	1	0	4
F	0	0	5

<Example 4> Impossible Results

Country	Win	Draw	Lose
A	5	0	0
B	3	1	1
C	2	1	2
D	2	0	3
E	0	0	5
F	1	0	4

Write a program that prints out 1 when the result is possible and 0 if not when there are total of 4 different results.
The maximum running time is 0.5 seconds. There are no partial credits.

Input Format (Input.txt)
Having four lines, each line has 18 numbers that are the results - wins, draws, and loses - of six countries in a same group with a space in between each numbers.

Output Format (Output.txt)
The output prints out 1 if the results are possible and 0 if not for 4 different results in one line with a space in between.

Example Input and Output

INPUT.TXT

```
5 0 0 3 0 2 2 0 3 0 0 5 4 0 1 1 0 4
4 1 0 3 0 2 4 1 0 1 1 3 0 0 5 1 1 3
5 0 0 4 0 1 2 2 1 2 0 3 1 0 4 0 0 5
5 0 0 3 1 1 2 1 2 2 0 3 0 0 5 1 0 4
```

OUTPUT.TXT

```
1 1 0 0
```

Stepping on Tiles

On the way to KOI venue, there are tiles lined up. Each tile has a natural number on it. Jason realized that the numbers on tiles are different and it was lined up in an increasing order. He tries to get to the venue by starting at a certain tile and stepping on tiles in order that increases with certain natural number.

Jason, however, realized that the number of tiles to step on consecutively depends on which tile to start and how much to increment each time to step on the different tile.

Jason decided to calculate the maximum sum of the natural numbers that were on the tiles that he stepped on the way to the venue as it was described the above. The tiles to step on consecutively need to be more than three.

For example, there are tiles with natural numbers lined up in the following way:

1, 2, 6, 7, 11, 12, 13, 15, 17, 20, 23

All the possible ways to step on more than three tiles, consecutively, are represented in the table below.

Increasing Amount	Order of Tiles	Sum
1	11, 12, 13	36
2	11, 13, 15, 17	56
3	17, 20, 23	60
4	7, 11, 15	33
5	2, 7, 12, 17	38
6	1, 7, 13	21
	11, 17, 23	51
7	6, 13, 20	39
8	7, 15, 23	45
9	2, 11, 20	33
11	1, 12, 23	36

Therefore, in this example, the possible maximum sum of natural numbers on tiles that Jason stepped on consecutively more than three times is 60 including the natural number on the staring tile.

Write a program that determines the maximum sum of the numbers on the tiles that Jason can step on consecutively more than three times, as the tiles are lines up in increasing order. If there are no more than three tiles that can be stepped on consecutively, print out 0.

The maximum running time is 0.3 seconds. There are no partial credits.

Input Format (Input.txt)

The first line has the number of tiles N ($3 \leq N \leq 3{,}000$). On the second line, there are natural numbers on tiles in an increasing order with a space in between the numbers. The natural number on each tile will be less than or equal to 1,000,000.

Output Format (Output.txt)

If there are three or more tiles for Jason to step on consecutively, print out the maximum sum on the first line. Otherwise, print out 0. The output always will be less than or equal to 2,000,000,000.

Example Input and Output

INPUT.TXT

```
11
1 2 6 7 11 12 13 15 17 20 23
```

OUTPUT.TXT

```
60
```

<Middle School Competition>

Grading

There are tests that need to be graded. The test has total of N number of questions. The points for questions from 1 to N corresponds to S_1, S_2, …, S_N. The points for each question are calculated in the following rule:

1. There will be no points for a wrong answer.
2. The points for number one are S_1 if the answer is right. If the i^{th} question answer is right, the points for that question depends on the correctness of the answer for $(i-1)^{th}$ question.
    a) If an answer to $(i-1)^{th}$ question is right, the points possible for i^{th} question becomes the original points for i^{th} question, S_i, added with points for $(i-1)^{th}$ question.
    b) If an answer to $(i-1)^{th}$ question is wrong, a point for i^{th} question will be S_i.
3. The total sum will be calculated by adding up the points from each question

For example, there is a test with total of nine questions and the possible points for each question are shown in Table 1.

<Table 1>

Question Number	1	2	3	4	5	6	7	8	9
Points	3	2	7	2	6	8	2	5	2

Table 2 shows whether a student got it right (O) or wrong (X) for each question.

<Table 2>

Question Number	1	2	3	4	5	6	7	8	9
Points	O	X	O	O	O	X	X	O	X

Then, the possible points for each question are shown below in Table 3. The total sum is now 39.

<Table 3>

Question Number	1	2	3	4	5	6	7	8	9
Points	3	0	7	9	15	0	0	5	0

There are numbers, however, that can’t be possible as a total sum such as 73, in this case with the points given in Table 1.

There will be a natural number K given with point for each question. Write a program that finds the smallest total sum M (K≤M) that is not possible.
The maximum running time is 1 second. There will be no partial credits.

Input Format (Input.txt)

The first line has a natural number N which represents the number of questions (1≤N≤150). The next line has point (1≤point≤100) for each question in order with a space in between. The natural number K (1≤K≤2,000,000,000) is given on the last line.

Output Format (Output.txt)

Print out the smallest total sum M (K≤M) that is not possible on the first line.

Example Input and Output

INPUT.TXT

```
9
3 2 7 2 6 8 2 5 2
72
```

OUTPUT.TXT

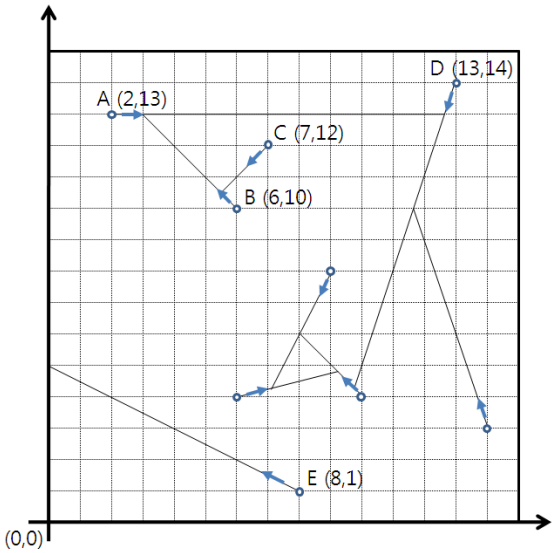
```
73
```

Snails

There are N number of snails inside a square that is fenced around. All snails start at the same time from their own starting points and move by the following rules listed below.

1. All snails have different starting points.
2. All snails have their own direction and only move straight from the direction they are given.
3. All snails have the velocity of 1cm per second.
4. Snails stop when they hit the fence.
5. When a snail reaches the path that is crossed by another snail, it stops.
6. If two or more snails reach at a point at the same time, they all stop.

The following picture shows the movement of nine snails. In the picture below, the side of a cell is 1cm.



In the picture above, snail C stopped as it reached the pathway of snail B, which stopped as it reached snail A's pathway. Snail E stopped as it reached the fence. Determine how long it takes for N number of snails to stop as each of them is given the starting point and direction by rounding up the time to two decimal places. As in the picture above, the last snail to stop is snail A and the total time for all snails to stop is 10.67 seconds.
The maximum running time is one second. There are no partial credits.

Input Format (Input.txt)

The first line has N which represents the total number of snails ($1 \le N \le 1{,}000$) and L which represents the length of one side of the fence in centimeter ($10 \le L \le 10{,}000$). The coordinate of lower left corner is (0, 0) and of upper right corner is (L, L) of the fence. For next N lines, each line has four natural numbers - x, y, p, q ($0 \le x, y, p, q \le L$) - that represent each snail's starting point and direction where

(x, y) represents the starting point and (p-x, q-y) represents the direction vector. Note that point (x, y) and point (p, q) can't be the same.

Output Format (Output.txt)

Print out the time that it took for all the snails to stop in the first line by rounding up to two decimal places. Be sure to print only up to two decimal places, for example, if the time is 10.667, 5, or 5.5, print out 10.67, 5.00, and 5.50, respectively.

Example Input and Output

INPUT.TXT

```
9 15
2 13 3 13
6 10 5 11
7 12 6 11
13 14 12 11
8 1 6 2
6 4 10 5
9 8 8 6
10 4 9 5
14 3 12 9
```

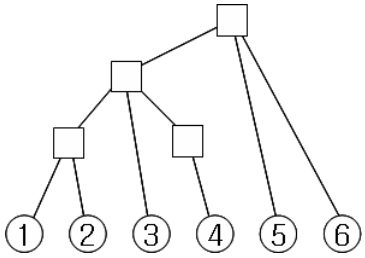
OUTPUT.TXT

```
10.67
```

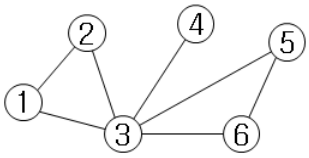
<High School Competition>

Phylogenetic Tree

In bioinformatics, it is very important to form a relationship among various biological species or other entities that are believed to have a common ancestor. Usually, this relation is represented as a phylogenetic tree. Each taxonomic unit is represented as a leaf node in the tree. In phylogenetic tree, the distance of a branch between two leaf nodes show how much they are related in evolutionary biology. The picture below shows the example of a phylogenetic tree.



Referring to the phylogenetic tree, we can define the graph that shows the closeness among the taxonomic units as in an explanation of evolutionary biology. This graph is called phylogenetic graph. *Similarity K* is defined as follows. The vertices of a graph represent each taxonomic unit. The edge of a graph exists if and only if the distance (the path length) of the nodes that are corresponding to the vertices in the phylogenetic tree needs to be less than or equal to K. The picture shown below is the *similarity 3* phylogenetic graph as it was defined from the phylogenetic tree above.



In the laboratory, after constructing a phylogenetic graph, the phylogenetic tree was lost. Therefore, the work needs to be done to reconstruct the phylogenetic tree.

Write a program that determines the phylogenetic tree when *similarity 3* phylogenetic tree is given. Note that the given phylogenetic graph is always connected and the corresponding phylogenetic tree always exists.

The maximum running time is one second. There are no partial credits.

Input Format (Input.txt)

The first line has a natural number N which represents vertices of a phylogenetic graph ($2 \le N \le 5,000$). The vertices are numbered from 1 to N. The second line has a natural number M that is edges of a phylogenetic graph ($1 \le M \le 1,000,000$). For M numbers of lines, each line has two numbers, v and w ($1 \le v, w \le 1,000$), that are the two vertices of an edge.

Output Format (Output.txt)

In the first line, print out S the number of edges in the phylogenetic tree. For each S number of lines, print out two vertices of an edge. Print out the edges in arbitrary order. The tree nodes need to be numbered from 1 to N for the sake of leaf nodes. Also, the numbers on the vertices in the phylogenetic graph need to be same on the leaf nodes in the phylogenetic tree. As for the numbers on other internal nodes, use N+1 to S+1 randomly. If there are more than one phylogenetic tree is possible, print out only one of them.

Example Input and Output

INPUT.TXT

```
6
7
1 2
2 3
3 1
3 4
3 5
5 6
6 3
```

OUTPUT.TXT

```
9
1 7
2 7
3 9
4 8
5 10
6 10
7 9
8 9
9 10
```