MIUP'16
MARATONA INTER-UNIVERSITÁRIA DE PROGRAMAÇÃO

http://www.di.fct.unl.pt/miup2016

Problem Set

Departamento de Informática da NOVA

15th October 2016

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

outsystems

Contents

Scientific Committee

- Alexandre Francisco (Instituto Superior Técnico, Universidade de Lisboa)
- André Restivo (Faculdade de Engenharia, Universidade do Porto)
- Carla Ferreira (Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa)
- Fábio Marques (Esc. Sup. de Tecnologia e Gestão de Águeda, Universidade de Aveiro)
- Filipe Araújo (Faculdade de Ciências e Tecnologia, Universidade de Coimbra)
- Hugo Sereno Ferreira (Faculdade de Engenharia, Universidade do Porto)
- José Saias (Escola de Ciências e Tecnologia, Universidade de Évora)
- Margarida Mamede (Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa)
- Paul Crocker (Faculdade de Engenharia, Universidade da Beira Interior)
- Pedro Guerreiro (Faculdade de Ciências e Tecnologia, Universidade do Algarve)
- Pedro Mariano (Faculdade de Ciências, Universidade de Lisboa)
- Pedro Ribeiro (Faculdade de Ciências, Universidade do Porto)
- Rui Mendes (Escola de Engenharia, Universidade do Minho)
- Simão Melo de Sousa (Faculdade de Engenharia, Universidade da Beira Interior)

Local Organizing Committee

- Anabela Duarte
- Artur Miguel Dias
- Bruno Machado
- João Costa Seco
- João Moura Pires
- Margarida Mamede
- Miguel Goulão
- Paulo Lopes
- Sandra Rainha
- Sérgio Duarte
- Susana Pereira
- Vítor Duarte

Compilers

Language	Compiler	Version	Command Line	Ext.
C	GCC	4.8.4	gcc -std=gnu11 -Wall -Wno-unused $file -lm	.c
C++	GCC	4.8.4	g++ -std=gnu++11 -Wall -Wno-unused $file -lm	.cpp
Java	OpenJDK	1.8.0	javac -encoding utf8 -classpath . $file	.java

Compilation Constraints

- **Maximum compilation time:** 60 seconds
- **Maximum source code size:** 100 Kb
- The source code for a problem must be submitted in a single file.
- In Java submissions, the `.java` file must have the same name as the class that contains the main method. That class must belong to the default package. There is no limit to the number of classes contained in the submitted file.

Execution Commands

Language	Command Line
C	./a.out
C++	./a.out
Java	java -Xmx256M -Xss256M -classpath . $name

Runtime Constraints (for all problems)

- **Maximum CPU time:** 2 seconds
- **Maximum static memory + heap:** 256 Mb
- **Maximum stack:** 256 Mb

Input/Output Format

- All lines (in the input and in the output) are ended by the newline character ('`\n`').
- Except when explicitly stated, single spaces are used as separators.
- No line starts or ends with any kind of whitespace.

Documentation

- man pages for C/C++ (using the command line)
- C++ STL documentation (link on Mooshak first page)
- Java SE Documentation (link on Mooshak first page)

Greedy Trading

CDFs[1] are instruments to speculate on financial markets. A CDF is a contract stipulating that a seller will pay to the buyer the difference between the current value of an asset and its value at contract time. CDFs are traded between individual traders and CDF providers. A CDF is started by making an opening trade on a particular instrument with the CDF provider. This creates a 'position' in that instrument, which can be either long (taking advantage of prices moving up) or short (taking advantage of prices moving down). There is no expiry date. Once the position is closed, the difference between the opening trade and the closing trade is paid as profit or loss.

Assuming that near real time access to stock quotes variation is available, your job is to help greedy traders by finding the sequences that maximize the value and their median. You should be able to do it also in near real time.

You should not be concerned that such information might not help traders at all. Let them play!

Task

Develop a tool that receives updates on price variation for a given CDF instrument and that is able to answer queries on the sequences with maximum sum and their median value. Your tool should be as fast as possible!

Input

A sequence of N lines starting either by `u`, `q` or `x`. Lines starting with `u` denote updates, where character `u` is followed by a single space and an integer number k. A line starting with `q` denotes a query command and a line starting with `x` should terminate the tool. All inputs start with an update command.

Constraints

$1 \leq N \leq 1\,000$ Number of commands

$-100 \leq k \leq 100$ Update values

[1] CDF – contract for difference.

Output

Only command `q` produces output. The output is a single line with three integer numbers:

- Two integers denoting the starting position and the end position of the non-empty contiguous subsequence with the maximum sum with respect to the sequence of numbers received until now. Assume that updates are numbered from position 0 (see the sample output below). Notice that if there are two or more such subsequences, then you should output information concerning the last subsequence. By last subsequence we mean the one that ends at a later position and, if more than one subsequence ends at the same position, then we want the one that starts at a later position.
- The third integer denotes the median of that subsequence. Notice that in case the sequence length is even you should output the greatest of the middle two values.

Sample Input

```
u -5
q
u 0
q
u 2
u 1
u 3
q
u 1
q
x
```

Sample Output

```
0 0 -5
1 1 0
2 4 2
2 5 2
```

Sample Explanation

We observed a sequence of updates for a given CDF instrument. Its value decreased by 5, but then it increased by 2, 1, 3 and 1. So, at the end, its value has increased by 2, but we are not interested on that. We want to answer queries q on non-empty subsequences with the maximum sum with respect to the sequence of updates till the query time. For the first query, we have that the subsequence with the maximum sum is -5 and its median is also -5, hence the output is `0 0 -5` where 0 is the index of the update instruction for value -5. For the second query we have the subsequence 0, hence the output is `1 1 0`. For the third query, we have that the subsequence with the maximum sum is $2, 1, 3$ and its median is 2, hence the output is `2 4 2` where 2 and 4 are the indexes of update instructions for values 2 and 3, respectively. For the last query, the subsequence with the maximum sum is $2, 1, 3, 1$ and its median is still 2, hence the output is `2 5 2`.

The Ant and the Grasshopper

Winter came and the grasshopper didn't have that much to eat. Unlike the ant of course... If at least it had collected something in the summer... It had to talk to the ant. Poor grasshopper. The ant had a bad reputation for speculation. It would certainly not give away anything. At most it would borrow some money!

And so it was. The ant lent it some money, but the grasshopper would have to pay everything back with interests. The ant would set the number of years for the reimbursement, and the grasshopper would pay a fixed amount along those years. The amount to pay in interests is recomputed every year, according to the amount still in debt at that moment in time. The difference between the fixed payment and the interests serves to amortize the debt. For example, assume that the Grasshopper borrows 100 at an interest rate of 5%, to be payed in 3 years. Then, we have the following payment plan (with values rounded to 3 decimal places):

Year	Debt	Payed	Interests	Amortization
1	100	36.721	5.000	31.721
2	68.279	36.721	3.414	33.307
3	34.972	36.721	1.749	34.972
Totals	-	110.163	10.163	100.000

Task

Your task is to compute the total amount the grasshopper will pay to the ant (interests plus amortization). You will receive the number of years, the amount borrowed and the interest rate (in percentage).

Input

The first line of the input is the number of instances to solve, n. In each of the following lines you will receive three values: the number of years, y, the amount borrowed, B, and the interest rate, r. The amount borrowed, B, and the interest rate, r, might not be integers.

Constraints

$1 \leq n \leq 1\,000$ Number of cases

$1 \leq y \leq 100$ Number of years

$0 < B \leq 1\,000$ Amount borrowed

$0 < r \leq 100$ Interest rate

Output

The output has one line per case, with the total amount payed to the Ant rounded to 3 decimal places.

Sample Input 1

```
1
3 100 5
```

Sample Output 1

```
110.163
```

Sample Input 2

```
4
20 100 6
15 1000 3
3 1.2 1.5
100 1000 100
```

Sample Output 2

```
174.369
1256.499
1.236
100000.000
```

Election of Representatives

The D'Hondt method is frequently used for allocation of parliamentary seats to parties, in proportion to the number of their received votes, balanced with minority representation. The representatives allocation algorithm is based on successive calculation, for each party, of quotients

$$q_i = \frac{v_i}{s_i + 1}$$

where v_i is the total number of votes in party i and s_i is the number of seats or representatives already allocated to that same party.

To begin, we calculate the quotients for each party with $s = 0$. At the end of each round we assign a seat to the party (k) with the highest quotient, so s_k is increased by 1. The first representatives place goes always to the most voted party. In the second iteration, s is 0 for all but the most voted, which means that its quotient value will be lower than before. Again, the party having the highest quotient in this round gets the next representative seat.

In case of a tie in the highest quotient value, on some round, the place is assigned to the least voted party in dispute. In this case, if there is still a tie in the number of votes of various parties, the seat is allocated to the party with the lowest index. The process ends when all seats are assigned.

Task

Implement a program to distribute parliamentary seats to parties, according to the election results and using the D'Hondt method as explained above.

Input

The first line has the total number of seats (S) for parliamentary representatives to be distributed. The second line has the number of parties (P). The following P lines, one for each party, have the number of votes (V) for that party. The party index, mentioned above, is an implicit integer starting from 1, with successive values from line to line.

Constraints

$1 \leq S \leq 250$	Number of seats
$1 \leq P \leq 20$	Number of parties
$0 \leq V \leq 1\,000\,000$	Number of votes for one party

Output

The program's output is made up of as many lines as there are parties. For each party, and arranged in identical order as in input, your program should write a line with the number of seats that were assigned to it.

Sample Input

```
3
4
20
12
8
2
```

Sample Output

```
2
1
0
0
```

Sample Explanation

To better understand this example, we can see the intermediate computations of each round in Table 1.

Party 1 has the highest quotient (in bold), so it receives the first representative seat. Quotients for rounds 2 and 3 are shown in the next lines. In round 2 the seat goes to party 2. And in the last round, the highest quotient appears again in party 1, resulting in its second seat.

After round 3 there are no more seats to assign, so the process ends with the output presented before, meaning that 2 seats were given to party 1, 1 seat to party 2, and zero to the remaining parties.

round	q_1	q_2	q_3	q_4	*party receiving a seat*
1	**20.0**	12.0	8.0	2.0	1
2	10.0	**12.0**	8.0	2.0	2
3	**10.0**	6.0	8.0	2.0	1

Table 1: Quotients and seat allocation in each round for the given example

The Modern Feud of the Capulets and Montagues

The tragedy involving the Capulets and Montagues would not have happened in today's age. Given that all members of both gangs have smartphones with GPS, it is easy for the families to know where their members are. In a recent truce, the families decided to trust a foreign company to produce an app that would always display the closest distance between any two members of both families.

Task

Due to the fact that this foreign company employs you as its personal slave, you have been entrusted with programing this app. You know that the modern city where this tragedy takes place is very geometric and, as such, all streets are orthogonal. Your program receives an entry for each person with its position on the 2D plane and the identifier of the family that person belongs to.

Your task is to print the minimum Manhattan distance between two members of the different families. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is:

$$|x_1 - x_2| + |y_1 - y_2|.$$

You may assume that the points are spread out evenly over the space.

Input

The input starts with an integer N that encodes the number of points. Each following line has 3 integers: X, Y, F encoding the point's coordinates and its family. There will be at least one point in each family.

Constraints

$2 \leq N \leq 100\,000$	Number of points
$0 \leq X \leq 1\,000\,000\,000$	X coordinate of a point
$0 \leq Y \leq 1\,000\,000\,000$	Y coordinate of a point
$0 \leq F < 2$	Family id

Output

Your program should print the minimum Manhattan distance between two points of different families.

Sample Input

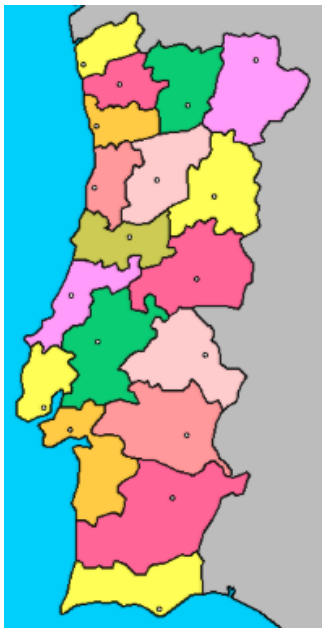
```
10
9 0 0
6 3 1
6 0 0
9 3 1
6 6 0
0 7 1
3 1 0
9 9 1
2 9 0
0 9 0
```

Sample Output

```
2
```

Administrative Reform

The Portuguese government is studying a new administrative reform: every district will be split into only two municipalities, each one with its capital city. Besides, any community (town, village or hamlet) must belong to the municipality whose capital is closest to it, taking into account all existing roads. Communities whose shortest distances to both capitals are equal can choose their new municipality.



For instance, suppose that the capitals of the Lisbon district are the city of Lisbon and Arruda dos Vinhos. The city of Lisbon has to be in the Lisbon municipality. But how can people from Amadora know their new municipality? Although there are many routes from Amadora to Lisbon, the shortest one has 9.8 km (via Estrada de Monsanto), whereas the shortest route from Amadora to Arruda dos Vinhos has 38.3 km (taking the A9). Therefore, Amadora must belong to the new Lisbon municipality.

Task

Write a program that, given the communities (towns, villages and hamlets) of a district, the existing roads between them, and the two new capitals, X and Y, computes how many communities must belong to the X municipality, how many communities must belong to the Y municipality, and how many communities are allowed to choose (between X and Y). It is guaranteed that there is some route between any two communities of the district. Besides, different roads share at most one endpoint in common.

Input

The first line of the input has two integers: C, which is the number of communities, and R, which is the number of roads. Communities are identified by integers, ranging from 0 to $C-1$.

Each of the following R lines contains three integers, c_1, c_2 and l, which indicate that there is a (two-way) road between the distinct communities c_1 and c_2 whose length is l.

The last line has two different integers, X and Y, which are the communities chosen to be the new capitals.

Constraints

$2 \leq C \leq 20\,000$	Number of communities
$1 \leq R \leq 150\,000$	Number of roads
$1 \leq l \leq 1\,000$	Length of a road

Output

The output has a single line with three integers: the number of communities that must belong to the X municipality, the number of communities that must belong to the Y municipality, and the number of communities that can choose their new municipality.

Sample Input 1

```
4 5
1 2 20
0 1 30
2 0 12
1 3 1
0 3 30
2 1
```

Sample Output 1

```
2 2 0
```

Sample Input 2

```
8 14
6 5 75
5 7 20
6 2 21
2 7 75
6 0 40
0 2 26
0 1 15
0 3 18
1 3 7
1 2 12
4 1 25
5 4 5
7 4 37
2 4 38
1 7
```

Sample Output 2

```
5 2 1
```

Rock-Me-Not

After a life of hard work, Joan managed to save just enough to retire and buy the house of her dreams. She loves flowers, so of course it has a large garden and she plans to fill it with *Forget-Me-Nots* – her favourite flowers in the world.

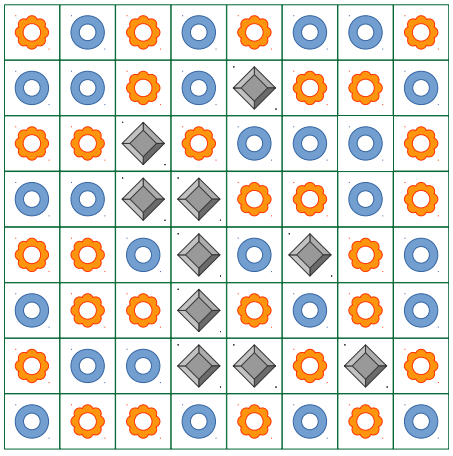
When she bought the house, the garden was packed with big rocks. She planned on removing them as soon as possible but when she tried to, she found out they were buried too deep and could not move them. She decided to embrace them and just create a zen garden by planting the flowers around them in a checkered pattern.

Forget-Me-Nots need lots of water, so she wants to have **two** sprinklers next to each flower. What is the maximum amount of flowers she can plant?

Task

Given a rectangular garden divided into equal squares, where some squares are occupied by unmovable rocks, calculate the maximum amount of flowers that can be planted knowing that each flower must have at least two sprinklers next to it (i.e. in an adjacent square – horizontally or vertically). Sprinklers and flowers need to be placed in their own square and cannot be placed in squares occupied by a rock. However, one sprinkler can be shared by several flowers.

The following figure depicts a valid configuration of flowers and sprinklers for the garden of sample 2. Flowers are drawn in orange, sprinklers in blue and rocks in dark grey.



Input

The first line of the input will contain two numbers, L and C, that represent the number of lines and columns of the garden in squared cells. The next L lines will each contain C characters. If the character is a '.' it represents an empty garden cell. If it is a '#' it represents a rock occupied cell.

Constraints

$1 \leq L \leq 9$ Length of the garden.
$1 \leq C \leq 9$ Width of the garden.

Output

A single number representing the maximum amount of flowers that can be planted.

Sample Input 1

```
3 3
...
...
...
```

Sample Output 1

```
5
```

Sample Input 2

```
8 8
........
....#...
..#.....
..##....
...#.#..
...#....
...##.#.
........
```

Sample Output 2

```
28
```

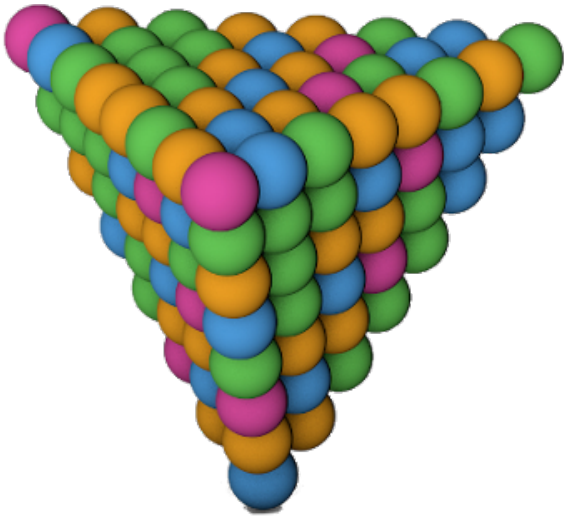
Eccentrics

I am sure you are aware of the EuroMillions lottery, the lottery that creates eccentric people. And perhaps you know how the draws are made. There's this huge transparent sphere with coloured balls inside, one ball for each number in the lottery. A strong stream of air is violently blown inside the sphere and the balls are made to fly like crazy, in all directions, until they are well shuffled. Then, the wind inside the sphere stops and one ball comes out, thus yielding a number. Afterwards, the wind starts again and the procedure repeats until the required number of balls are extracted.

This is all fine, but recently the Society for the Protection of Lottery Balls launched a protest movement, claiming that the balls are subject to excessive stress when they have to tumble inside the sphere, pushed by the irrational stream of air. They suggest an alternative, more gentle procedure.

Instead of a windy sphere, a static inverted tetrahedron is used. The base of the tetrahedron, which is facing upwards, is open. The balls are to be dropped simultaneously from a height, in such a way that they fall inside the tetrahedron. This, the Society claims, ensures enough randomness, and it seems the balls actually enjoy free falling. Then, the tip of the tetrahedron (which is pointing downwards) opens, allowing for just one ball to come out. That's the ball that was at the bottom tip of the tetrahedron, of course. As the first ball comes out, one of the balls in the second layer moves down, filling the empty space, so to speak. And a ball from the third layer moves down, to fill the empty space left by the ball from the second layer that moved down. And so on, up to the top layer of balls.

The following figure illustrates the original configuration of the inside of the tetrahedron after 120 coloured balls have been randomly dropped into it, filling exactly eight layers:



Your task is to prove by computer that this system is biased. In fact, physicists have pointed out that, all balls being made with the same material, heavier balls glide down first.

More precisely, when an empty space is created because a ball has moved down, of all the balls than can fill that empty space (there can be zero, one, two or three such balls, on the layer above the layer of the empty space that was just created), gravity will choose the heaviest, provided there is at least one.

No ties will ever occur, because it is impossible to produce balls with exactly the same weight.

Task

Write a program that, given a description of the balls inside the tetrahedron, computes the sequence of balls falling out from the bottom tip of the tetrahedron, once the tip opens.

Input

The first line of the input contains one positive integer, N, representing the number of layers of balls inside the tetrahedron. All layers are full.

Each layer S, for $S \in \{1, 2, \ldots, N\}$, is represented by S consecutive input lines of positive integers, in a triangular pattern: the first line has one number, the second line, two numbers, etc., and the S-th line, S numbers. Each number represents the weight of a ball and there are no duplicate numbers. Hence, the weights uniquely represent the balls.

Each triangle of numbers, in the input, represents graphically the arrangement of balls in the respective layer.

Constraints

$0 < N \leq 16$ Number of layers

$0 < W < 2^{31}$ Weight of a ball

Output

The output shall contain one line for each ball, in the order they fall out from the bottom of the tetrahedron.

(Continues on the next page)

Sample Input

```
3
9
1
8 7
5
2 3
4 10 11
```

Sample Output

```
9
8
10
7
11
4
3
2
1
5
```

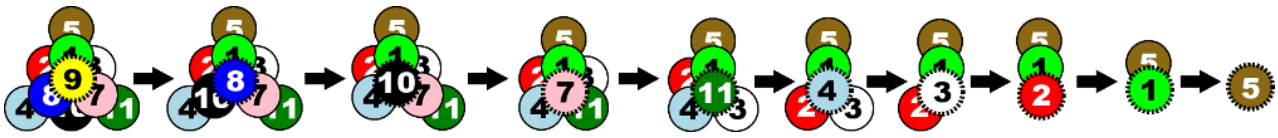
Sample Explanation

There are three layers. The first ball to fall out is the ball at the bottom, 9. All three balls at the second layer could, in principle, move down to fill the space left empty when ball 9 fell out. The heaviest of the three is ball 8. Hence, this one moves down, to the place which previously was occupied by ball 9. The balls on the third layer that could move down to the place vacated when ball 8 descended are 2, 4 and 10. The heaviest is 10. This is the one that descends. The third layer is the top layer, so there are no balls to fill the place vacated by ball 10.

The bottom ball is now 8. It falls out. On the second layer we have balls 1, 7 and 10. Hence 10, being the heaviest, descends. The balls on the third layer that could descend to fill the place vacated by ball 10 are balls 2 and 4. The heaviest is 4. This is the ball that descends, to fill the place previously occupied by ball 10.

Next ball to fall out is 10. On the second layer, we now have 1, 4 and 7. The heaviest is 7. It comes down. On the third layer, the balls that could now descend are 3 and 11. Hence, 11 descends.

And so on. The full story is depicted in the following figure, in which we are looking at the tetrahedron from below:



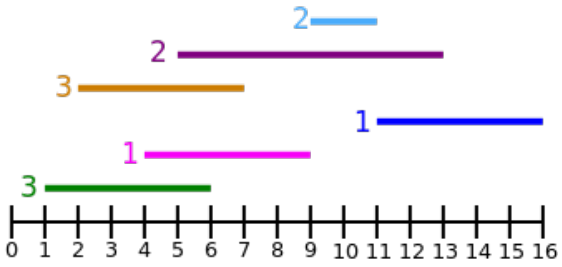
(This page is intentionally left blank)

Problem Setters

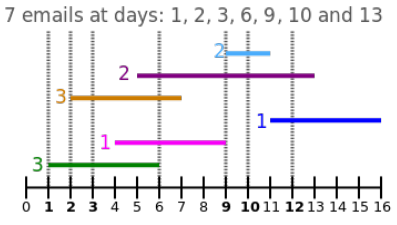
Amy is organizing a programming contest and she is the leader of the scientific committee, whose main responsibility is the creation of the problem set. Amy has a strict deadline for the initial proposal of problems but she knows that without sending emails remembering the deadline some of the members of the committee will forget about the contest. She wants to send the fewest possible number of emails, but she also knows that not all dates are equally effective for this task: send a message too soon and some of the members will forget about the contest; send a message too late and some will not have enough time. Since Amy knows all members very well from previous contests, she can estimate what is the minimum number of emails each person should receive and in which interval of time she should reach each of them.

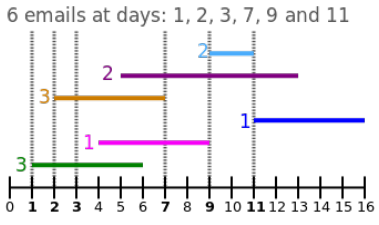
Imagine for instance the following scenario with six members of the committee (besides Amy) and their respective minimum number of emails and intervals:

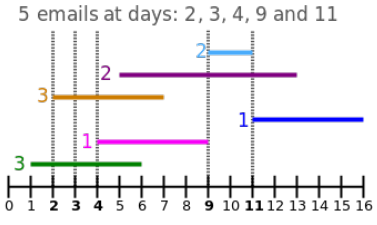
Name	Min Emails	Interval
Steve	(2)	[9, 11]
Edna	(2)	[5, 13]
Mary	(3)	[2, 7]
James	(1)	[11,16]
Barbara	(1)	[4, 9]
David	(3)	[1 ,6]



Every time Amy sends an email, all committee members will be put as recipients. Knowing that she will never send more than one email per day, she needs to decide on which days she should send a message. The figure below shows three different valid possibilities, all of them respecting both the minimum number of emails and intervals of each person. Out of these three possibilities Amy prefers the last one, which requires less emails than the other two.







Note that the last alternative is an optimal solution because, in this example (which corresponds to sample 1), Amy needs to send at least 5 emails.

Can you help Amy decide what is the minimum number of emails needed?

Task

Given N k_i values and intervals $[a_i, b_i]$, indicating that the i-th member of the committee needs to receive a minimum of k_i emails between days a_i and b_i (inclusive) before the deadline, your task is to determine the smallest number of days in which Amy needs to send an email so that every member will receive at least k_i emails on its corresponding interval.

Input

The first line of the input contains N, the number of members of the committee. The following N lines contain each one three integer values, k_i, a_i and b_i, indicating the minimum amount of emails (k_i) and interval $[a_i, b_i]$ for the i-th committee member.

Constraints

$1 \leq N \leq 35\,000$	Number of members of the scientific committee
$1 \leq k_i \leq 5$	Minimum amount of emails per person
$1 \leq a_i \leq b_i \leq 10^9$	Days of each interval (with $b_i - a_i + 1 \geq k_i$)

Output

The output should consist of a single line with an integer indicating the minimum number of emails needed.

Sample Input 1

```
6
2 9 11
2 5 13
3 2 7
1 11 16
1 4 9
3 1 6
```

Sample Output 1

```
5
```

Sample Input 2

```
8
1 20 20
2 5 7
1 7 9
2 4 8
1 3 5
1 2 3
1 9 10
4 15 18
```

Sample Output 2

```
9
```

Surveillance

Samuel is starting a business in the surveillance industry. He provides closed IPTV surveillance systems to stores. He aims to provide a reliable service but with a competitive price. To achieve this he decided to add some limitations to the system:

1. The system only allows one surveillance camera.
2. The camera can only be set at store corners.

Despite the budget limitations, Samuel also decided that he would use cameras with the ability to capture video in 360°.

Even with a functional system, Samuel continues with an unsolved problem: how to determine in which corner should the IPTV camera be installed? The only requirement is to install the camera in the corner that covers more area.

Samuel asked you to implement a solution that would help him. Each store can be represented by a simple polygon (whose boundary does not self-intersect), and each corner by a vertex. Consider that the polygons have no holes, meaning that nothing inside the store blocks the visibility of the camera, except an exterior wall. Observe the figure, which represents the data of sample 1.

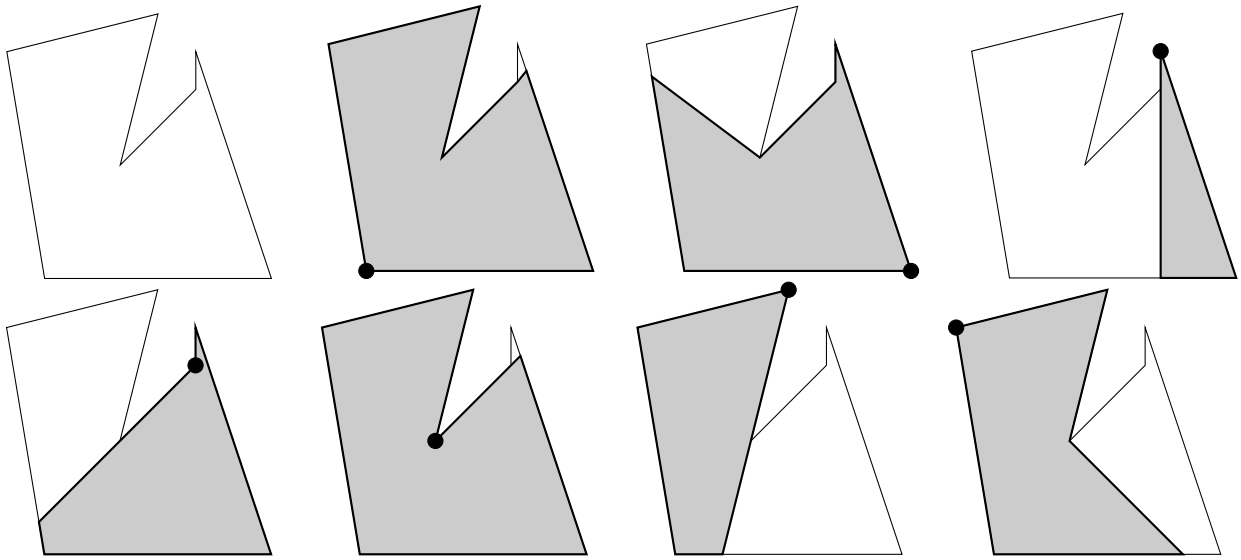


Figure 1: Store representation and visibility polygons of each vertex

Task

Help Samuel to find the store corner where to install the camera.

Input

The first line of the input has an integer n which indicates the number of vertices of the polygon that represents the store. The following n lines contain the coordinates x, y of the vertices of the polygon, listed in counterclockwise order. The coordinates' values are integers and are separated by a space.

Constraints

$3 \leq n \leq 50$ Number of vertices

$0 \leq x \leq 500$ x coordinate of a vertex

$0 \leq y \leq 500$ y coordinate of a vertex

Output

The output has a single line with the coordinates, separated by a space, of the vertex that covers the largest area of the polygon, hence the one where the camera will be installed. In case the largest area is covered by two or more vertices, choose the vertex closest to the origin (w.r.t. the Euclidean distance). If there is still a tie (i.e., several vertices cover areas of largest size and are all at the same distance from the origin), choose the one closest to the x-axis.

Sample Input 1

```
7
1 0
7 0
5 6
5 5
3 3
4 7
0 6
```

Sample Output 1

```
1 0
```

Sample Input 2

```
4
0 0
1 0
1 1
0 1
```

Sample Output 2

```
0 0
```