School Personal Contest #1 (Winter Computer School 2010/11)<br/>-<br/>Codeforces Beta Round #38 (ACM-ICPC Rules)

# A. Army

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The Berland Armed Forces System consists of $n$ ranks that are numbered using natural numbers from $1$ to $n$, where $1$ is the lowest rank and $n$ is the highest rank.

One needs exactly $d_i$ years to rise from rank $i$ to rank $i+1$. Reaching a certain rank $i$ having not reached all the previous $i-1$ ranks is impossible.

Vasya has just reached a new rank of $a$, but he dreams of holding the rank of $b$. Find for how many more years Vasya should serve in the army until he can finally realize his dream.

## Input

The first input line contains an integer $n$ ($2 \le n \le 100$). The second line contains $n-1$ integers $d_i$ ($1 \le d_i \le 100$). The third input line contains two integers $a$ and $b$ ($1 \le a < b \le n$). The numbers on the lines are space-separated.

## Output

Print the single number which is the number of years that Vasya needs to rise from rank $a$ to rank $b$.

**Sample test(s)**

input

```
3
5 6
1 2
```

output

```
5
```

input

```
3
5 6
1 3
```

output

```
11
```

# B. Chess

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two chess pieces, a rook and a knight, stand on a standard chessboard $8 \times 8$ in size. The positions in which they are situated are known. It is guaranteed that none of them beats the other one.

Your task is to find the number of ways to place another knight on the board so that none of the three pieces on the board beat another one. A new piece can only be placed on an empty square.

### Input

The first input line contains the description of the rook's position on the board. This description is a line which is 2 in length. Its first symbol is a lower-case Latin letter from $\mathrm{a}$ to $\mathrm{h}$, and its second symbol is a number from 1 to 8. The second line contains the description of the knight's position in a similar way. It is guaranteed that their positions do not coincide.

### Output

Print a single number which is the required number of ways.

#### Sample test(s)

input

```
a1
b2
```

output

```
44
```

input

```
a8
d4
```

output

```
38
```

# C. Blinds

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The blinds are known to consist of opaque horizontal stripes that can be rotated thus regulating the amount of light flowing in the room. There are $n$ blind stripes with the width of 1 in the factory warehouse for blind production. The problem is that all of them are spare details from different orders, that is, they may not have the same length (it is even possible for them to have different lengths)

Every stripe can be cut into two or more parts. The cuttings are made perpendicularly to the side along which the length is measured. Thus the cuttings do not change the width of a stripe but each of the resulting pieces has a lesser length (the sum of which is equal to the length of the initial stripe)

After all the cuttings the blinds are constructed through consecutive joining of several parts, similar in length, along sides, along which length is measured. Also, apart from the resulting pieces an initial stripe can be used as a blind if it hasn't been cut. It is forbidden to construct blinds in any other way.

Thus, if the blinds consist of $k$ pieces each $d$ in length, then they are of form of a rectangle of $k \times d$ bourlemeters.

Your task is to find for what window possessing the largest possible area the blinds can be made from the given stripes if on technical grounds it is forbidden to use pieces shorter than $l$ bourlemeter. The window is of form of a rectangle with side lengths as positive integers.

### Input

The first output line contains two space-separated integers $n$ and $l$ ($1 \le n, l \le 100$). They are the number of stripes in the warehouse and the minimal acceptable length of a blind stripe in bourlemeters. The second line contains space-separated $n$ integers $a_i$. They are the lengths of initial stripes in bourlemeters ($1 \le a_i \le 100$).

### Output

Print the single number — the maximal area of the window in square bourlemeters that can be completely covered. If no window with a positive area that can be covered completely without breaking any of the given rules exist, then print the single number $0$.

**Sample test(s)**

input

```
4 2
1 2 3 4
```

output

```
8
```

input

```
5 3
5 5 7 3 1
```

output

```
15
```

input

```
2 3
1 2
```

output

```
0
```

### Note

In the first sample test the required window is $2 \times 4$ in size and the blinds for it consist of 4 parts, each 2 bourlemeters long. One of the parts is the initial stripe with the length of 2, the other one is a part of a cut stripe with the length of 3 and the two remaining stripes are parts of a stripe with the length of 4 cut in halves.

# D. Vasya the Architect

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once Vasya played bricks. All the bricks in the set had regular cubical shape. Vasya vas a talented architect, however the tower he built kept falling apart.

Let us consider the building process. Vasya takes a brick and puts it on top of the already built tower so that the sides of the brick are parallel to the sides of the bricks he has already used. Let's introduce a Cartesian coordinate system on the horizontal plane, where Vasya puts the first brick. Then the projection of brick number $i$ on the plane is a square with sides parallel to the axes of coordinates with opposite corners in points $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$. The bricks are cast from homogeneous plastic and the weight of a brick $a \times a \times a$ is $a^3$ grams.

It is guaranteed that Vasya puts any brick except the first one on the previous one, that is the area of intersection of the upper side of the previous brick and the lower side of the next brick is always positive.

We (Vasya included) live in a normal world where the laws of physical statics work. And that is why, perhaps, if we put yet another brick, the tower will collapse under its own weight. Vasya puts the cubes consecutively one on top of the other until at least one cube loses the balance and falls down. If it happens, Vasya gets upset and stops the construction. Print the number of bricks in the maximal stable tower, that is the maximal number $m$ satisfying the condition that all the towers consisting of bricks 1, 2, ..., $k$ for every integer $k$ from 1 to $m$ remain stable.

### Input

The first input file contains an integer $n$ $(1 \le n \le 100)$ which is the number of bricks. Each of the next $n$ lines contains four numbers $x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2}$ $(x_{i,1} \ne x_{i,2}, |x_{i,1} - x_{i,2}| = |y_{i,1} - y_{i,2}|)$ which are the coordinates of the opposite angles of the base of the brick number $i$. The coordinates are integers and their absolute value does not exceed $50$.

The cubes are given in the order Vasya puts them. It is guaranteed that the area of intersection of the upper side of the brick number $i - 1$ and the lower side of the brick number $i$ is strictly strictly greater than zero for all $i \ge 2$.

### Output

Print the number of bricks in the maximal stable tower.

#### Sample test(s)

input

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

output

```
2
```

input

```
2
0 0 3 3
2 0 5 3
```

output

```
1
```

input

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

output

```
3
```

# E. Let's Go Rolling!

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

On a number axis directed from the left rightwards, $n$ marbles with coordinates $x_1, x_2, ..., x_n$ are situated. Let's assume that the sizes of the marbles are infinitely small, that is in this task each of them is assumed to be a material point. You can stick pins in some of them and the cost of sticking in the marble number $i$ is equal to $c_i$, number $c_i$ may be negative. After you choose and stick the pins you need, the marbles will start to roll left according to the rule: if a marble has a pin stuck in it, then the marble doesn't move, otherwise the marble rolls all the way up to the next marble which has a pin stuck in it and stops moving there. If there is no pinned marble on the left to the given unpinned one, it is concluded that the marble rolls to the left to infinity and you will pay an infinitely large fine for it. If no marble rolled infinitely to the left, then the fine will consist of two summands:

- the sum of the costs of stuck pins;
- the sum of the lengths of the paths of each of the marbles, that is the sum of absolute values of differences between their initial and final positions.

Your task is to choose and pin some marbles in the way that will make the fine for you to pay as little as possible.

### Input

The first input line contains an integer $n$ ($1 \le n \le 3000$) which is the number of marbles. The next $n$ lines contain the descriptions of the marbles in pairs of integers $x_i$, $c_i$ ($-10^9 \le x_i, c_i \le 10^9$). The numbers are space-separated. Each description is given on a separate line. No two marbles have identical initial positions.

### Output

Output the single number — the least fine you will have to pay.

**Sample test(s)**

input

```
3
2 3
3 4
1 2
```

output

```
5
```

input

```
4
1 7
3 1
5 10
6 1
```

output

```
11
```

# F. Smart Boy

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Once Petya and Vasya invented a new game and called it "Smart Boy". They located a certain set of words — the dictionary — for the game. It is admissible for the dictionary to contain similar words.

The rules of the game are as follows: first the first player chooses any letter (a word as long as $1$) from any word from the dictionary and writes it down on a piece of paper. The second player adds some other letter to this one's initial or final position, thus making a word as long as $2$, then it's the first player's turn again, he adds a letter in the beginning or in the end thus making a word as long as $3$ and so on. But the player mustn't break one condition: the newly created word must be a substring of a word from a dictionary. The player who can't add a letter to the current word without breaking the condition loses.

Also if by the end of a turn a certain string $s$ is written on paper, then the player, whose turn it just has been, gets a number of points according to the formula:

$$score(s) = \left(\sum_{i=1}^{|s|} value(s_i)\right) \cdot \max_{1 \le i \le |s|} \{value(s_i)\} + num(s)$$

where

- $value(c)$ is a sequence number of symbol $c$ in Latin alphabet, numbered starting from $1$. For example, $value(\mathtt{a}) = 1$, and $value(\mathtt{z}) = 26$.
- $num(s)$ is the number of words from the dictionary where the line $s$ occurs as a substring at least once.

Your task is to learn who will win the game and what the final score will be. Every player plays optimally and most of all tries to win, then — to maximize the number of his points, then — to minimize the number of the points of the opponent.

### Input

The first input line contains an integer $n$ which is the number of words in the located dictionary $(1 \le n \le 30)$. The $n$ lines contain the words from the dictionary — one word is written on one line. Those lines are nonempty, consisting of Latin lower-case characters no longer than $30$ characters. Equal words can be in the list of words.

### Output

On the first output line print a line "First" or "Second" which means who will win the game. On the second line output the number of points of the first player and the number of points of the second player after the game ends. Separate the numbers by a single space.

**Sample test(s)**

input

```
2
aba
abac
```

output

```
Second
29 35
```

input

```
3
artem
nik
max
```

output

```
First
2403 1882
```

# G. Queue

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

On a cold winter evening our hero Vasya stood in a railway queue to buy a ticket for Codeforces championship final. As it usually happens, the cashier said he was going to be away for 5 minutes and left for an hour. Then Vasya, not to get bored, started to analyze such a mechanism as a queue. The findings astonished Vasya.

Every man is characterized by two numbers: $a_i$, which is the importance of his current task (the greater the number is, the more important the task is) and number $c_i$, which is a picture of his conscience. Numbers $a_i$ form the permutation of numbers from $1$ to $n$.

Let the queue consist of $n - 1$ people at the moment. Let's look at the way the person who came number $n$ behaves. First, he stands at the end of the queue and the does the following: if importance of the task $a_i$ of the man in front of him is less than $a_n$, they swap their places (it looks like this: the man number $n$ asks the one before him: "Erm... Excuse me please but it's very important for me... could you please let me move up the queue?"), then he again poses the question to the man in front of him and so on. But in case when $a_i$ is greater than $a_n$, moving up the queue stops. However, the man number $n$ can perform the operation no more than $c_n$ times.

In our task let us suppose that by the moment when the man number $n$ joins the queue, the process of swaps between $n - 1$ will have stopped. If the swap is possible it necessarily takes place.

Your task is to help Vasya model the described process and find the order in which the people will stand in queue when all the swaps stops.

### Input

The first input line contains an integer $n$ which is the number of people who has joined the queue ($1 \le n \le 10^5$). In the next $n$ lines descriptions of the people are given in order of their coming — space-separated integers $a_i$ and $c_i$ ($1 \le a_i \le n$, $0 \le c_i \le n$). Every description is located on s single line. All the $a_i$'s are different.

### Output

Output the permutation of numbers from $1$ to $n$, which signifies the queue formed according to the above described rules, starting from the beginning to the end. In this succession the $i$-th number stands for the number of a person who will stand in line on the place number $i$ after the swaps ends. People are numbered starting with $1$ in the order in which they were given in the input. Separate numbers by a space.

**Sample test(s)**

input

```
2
1 0
2 1
```

output

```
2 1
```

input

```
3
1 3
2 3
3 3
```

output

```
3 2 1
```

input

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

output

```
3 1 5 4 2
```

# H. The Great Marathon

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

On the Berland Dependence Day it was decided to organize a great marathon. Berland consists of $n$ cities, some of which are linked by two-way roads. Each road has a certain length. The cities are numbered from 1 to $n$. It is known that one can get from any city to any other one by the roads.

$n$ runners take part in the competition, one from each city. But Berland runners are talkative by nature and that's why the juries took measures to avoid large crowds of marathon participants. The jury decided that every runner should start the marathon from their hometown. Before the start every sportsman will get a piece of paper containing the name of the city where the sportsman's finishing line is. The finish is chosen randomly for every sportsman but it can't coincide with the sportsman's starting point. Several sportsmen are allowed to finish in one and the same city. All the sportsmen start simultaneously and everyone runs the shortest route from the starting point to the finishing one. All the sportsmen run at one speed which equals to $1$.

After the competition a follow-up table of the results will be composed where the sportsmen will be sorted according to the nondecrease of time they spent to cover the distance. The first $g$ sportsmen in the table will get golden medals, the next $s$ sportsmen will get silver medals and the rest will get bronze medals. Besides, if two or more sportsmen spend the same amount of time to cover the distance, they are sorted according to the number of the city where a sportsman started to run in the ascending order. That means no two sportsmen share one and the same place.

According to the rules of the competition the number of gold medals $g$ must satisfy the inequation $g_1 \le g \le g_2$, where $g_1$ and $g_2$ are values formed historically. In a similar way, the number of silver medals $s$ must satisfy the inequation $s_1 \le s \le s_2$, where $s_1$ and $s_2$ are also values formed historically.

At present, before the start of the competition, the destination points of every sportsman are unknown. However, the press demands details and that's why you are given the task of counting the number of the ways to distribute the medals. Two ways to distribute the medals are considered different if at least one sportsman could have received during those distributions different kinds of medals.

### Input

The first input line contains given integers $n$ and $m$ ($3 \le n \le 50$, $n - 1 \le m \le 1000$), where $n$ is the number of Berland towns and $m$ is the number of roads.

Next in $m$ lines road descriptions are given as groups of three integers $v$, $u$, $c$, which are the numbers of linked towns and its length ($1 \le v, u \le n$, $v \ne u$, $1 \le c \le 1000$). Every pair of cities have no more than one road between them.

The last line contains integers $g_1$, $g_2$, $s_1$, $s_2$ ($1 \le g_1 \le g_2$, $1 \le s_1 \le s_2$, $g_2 + s_2 < n$). The input data numbers, located on one line, are space-separated.

### Output

Print the single number — the number of ways to distribute the medals. It is guaranteed that the number fits in the standard 64-bit signed data type.

**Sample test(s)**

input

```
3 2
1 2 1
2 3 1
1 1 1 1
```

output

```
3
```

input

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

output

```
19
```

input

```
3 3
1 2 2
2 3 1
3 1 2
1 1 1 1
```

output

```
4
```