Running sushi

Description

Alexander and his friends are visiting a new running sushi restaurant. After the eating, they sit in front of huge piles of coloured plates. In the running sushi restaurant, each plate is priced according to its colour. To predict the amount to pay, they want to use a program to compute the total price in function of the plate colours, the quantities of plates and the plate prices of each colour. Since it is the inauguration day, every tenth plate of the same colour is free.

Task

Write the program for Alexander and his friends.

Constraints

- If $\boldsymbol{C}$ is the number of existing plate colours, $\boldsymbol{C} \in \mathbb{N}$ and $1 \leq \boldsymbol{C} \leq 50$.
- If $\boldsymbol{N_i}$ is the quantity of plates of colour $\boldsymbol{i}$, $\boldsymbol{N_i} \in \mathbb{N}$ and $1 \leq \boldsymbol{N_i} \leq 1000$.
- If $\boldsymbol{P_i}$ is the price of one plate of colour $\boldsymbol{i}$, $\boldsymbol{P_i} \in \mathbb{N}$ and $1 \leq \boldsymbol{P_i} \leq 100$.

Input and output of the program

Input data

The input consists of $\boldsymbol{C} + 1$ lines. The first line indicates the value of $\boldsymbol{C}$. In the next $\boldsymbol{C}$ lines, each line consists of three values: the color $\boldsymbol{i}$ (indicated in one word), the quantity of plates of that color and the price of one plate of that color. These three values are separated by a space.

Output data

The output is the total price to pay for all plates

Execution example

Input file	Output file (result)
9 red 7 4 darkblue 6 6 lightblue 2 5 orange 3 5 white 18 1 pink 5 3 green 0 7 black 1 10 yellow 8 1	139

Technical constraints

Task name	sushi
Input file	standard input
Output file	standard output
Time limit	0,1 second
Memory limit	256 megabytes

Poll star

Description

An imaginary country is subdivided into electoral districts.

In each district, voters cast their votes in polling stations.

Each polling station validates only the votes of the winning candidate, the votes of the other candidates being ignored.

Task

You must provide the program they need to determine in each district the **Star** (elected with a very large majority).

We consider as the **Star** the one who was able to get more than twice the total number of votes validated for any other candidate in the same district.

Example

There are two districts. The first one has five, the second four polling stations.

In the first district, Candidate 1 received a total of 800 votes, Candidate 2 received only 230 votes. Since Candidate 1 received more than double the number of votes of Candidate 2, Candidate 1 can be considered as being the **Star**.

In the second district, candidate 1 received a total of 100 votes, candidate 2 received 130 votes and candidate 3 received 90 votes. Since none of the candidates has more than double the votes of any other candidate, none of them is the **Star**.

Constraints

- If D is the number of districts: $1 \leq D \leq 100$.
- If C is the number of candidates in a district: $2 \leq C \leq 50$.
- If P is the number of polling stations in a district: $1 \leq P \leq 1000$.
- The number of votes in each polling station: $1 \leq V \leq 1000$.

Input and output of the program

Input data

The first line contains the number of districts ***D***.

For each district we have:

- One line with two integers. The first integer indicates the number of candidates **C**, the second indicates the number of polling stations ***P***.
- ***P*** lines – one per polling station – indicating each the index number of a candidate and the number of votes that candidate has received, separated by a space.

Output data

For each district the program gives:

- A first line with the text “District“ followed by the index number of the district and separated by a space.
- A second line with either the index number of the “Star” candidate of that district or the text “No Star in this district”.

Execution example

Input file	Output file (result)
2	District 1
2 5	1
1 100	District 2
1 300	No Star in this district
2 80	
2 150	
1 400	
3 4	
3 90	
2 50	
1 100	
2 80	

Distribution of points

Subtask	Points	Constraints/Description
1	10	**C** = 2
2	10	No additional constraints

Technical constraints

Task name	star
Input file	standard input
Output file	standard output
Time limit	0,1 second
Memory limit	256 megabytes

Flipped numbers

Description

Lea loves numbers that read the same way after being flipped 180 degrees. For instance, **609** is such a number, because if we flip it by 180° it is still **609**. She realises that in the decimal system, the only digits that can be used to construct such a number are **0**, **1**, **6**, **8**, **9** because if we flip these digits, we obtain: **0** -> **0**, **1** -> **1**, **6** -> **9**, **8** -> **8**, **9** -> **6**. In binary, all digits have this property: **0** -> **0**, **1** -> **1** while in the hexadecimal system only the digits **3** -> **E**, **E** -> **3** are added, while digits **A**, **B**, **C**, **D**, **F** do not make another number while flipping.

Task

Lea gives you two numbers ***X*** and ***N***. If ***X*** = 2, output all binary numbers that have ***N*** digits (without leading zeros) in numerical order. If ***X*** = 10, do the same for decimal numbers and if ***X*** = 16, do the same for hexadecimal numbers.

Example

If ***X*** = 10 and ***N*** = 1, the only numbers that satisfy the flipping properties are **0**, **1**, **8**. Since **6** maps to **9** and **9** to **6** they do not qualify here.

Constraints

- $X \in \{2; 10; 16\}$
- $0 < N \leq 10$

Input and output of the program

Input data

Two space separated numbers ***X*** and ***N***.

Output data

All numbers with the required properties sorted in numerical order, each number on a new line.

Execution examples

Input file	Output file (result)

`2 4`	`1001`
	`1111`

Input file	Output file (result)

`16 2`	`11`
	`3E`
	`69`
	`88`
	`96`
	`E3`

Distribution of points

Subtask	Points	Constraints/Description
1	5	Lea is only interested in binary numbers. X = 2 for all testcases.
2	10	Lea is only interested in decimal numbers. X = 10 for all testcases.
3	10	Lea is only interested in hexadecimal numbers. X = 16 for all testcases.
4	5	No constraints.

Technical constraints

Task name	flipped_numbers
Input file	standard input
Output file	standard output
Time limit	1 second
Memory limit	256 megabytes

Enemy detectors

Description

To survey a square grid, enemy detectors were placed at various positions. Each detector surveys all cells that are in the same row, column or diagonal, including its own cell. Unfortunately, the detectors were placed haphazardly, so that (most often) they do not survey every cell of the grid, and some detectors might even be at the same position.

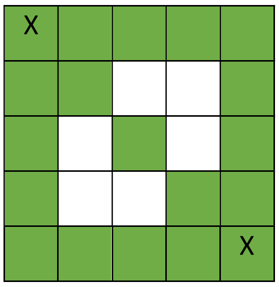
Task

Given a grid size ***N*** and ***k*** detectors with their respective coordinates (x_i, y_i), compute the number of cells that are not surveyed by detectors.

Example

Let's assume ***N*** = 5 and ***k*** = 2, with detectors at positions (1, 1) and (5, 5).

As you can see, 6 cells are not covered by the placed detectors.



Constraints

- $0 < N \leq 10^4$
- $0 < k \leq 1000$
- $1 \leq x_i, y_i \leq N$

Input and output of the program

Input data

The first line contains ***N*** (the size of the grid) and ***k*** (the number of detectors), separated by a space.

The ***k*** following lines contain x_i and y_i (the coordinates of the different detectors), separated by a space.

All these numbers are positive integers.

Output data

The number of cells not surveyed by the detectors.

Execution example

Input file	Output file (result)

`5 2`
`1 1`
`5 5`
	`6`

Distribution of points

Subtask	Points	Constraints/Description
1	10	***k*** = 1
2	10	***N*** ≤ 100
3	20	No additional constraints.

Technical constraints

Task name	detectors
Input file	standard input
Output file	standard output
Time limit	1 second
Memory limit	256 megabytes