Problem A: Chocolate

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Martin won the programming championship at his university. His prize is going to be a bar of chocolate. He and his University are able to choose the prize among N different brands of chocolate. Chocolate bars of different brands always have different prices. Moreover, every chocolate brand has an ancestor, except the brand "Router's Ports". The ancestor of a brand can obviously only be a brand produced chronologically earlier. Historically, the first chocolate brand was "Router's Ports".

The chief accountant of the university and Martin both have a say in choosing the brand of the chocolate Martin gets. Of course, Martin's goal is to choose a more expensive one while the accountant wants the university to save some money. The process of choosing starts from "Router's Ports". They both take turns at picking the brand they're (currently) in favor of, until they finally agree on one. Martin can either select the current chocolate or one of its descendants (if there is one). The accountant naively assumes that all new chocolates are worse and therefore cheaper, that's why he always chooses a descendant of the current one, as long as there is one. Hence they can only agree on a chocolate with no descendants. Martin starts the choosing process. Find out the most expensive chocolate bar Martin can get as a result of this process (if neither one chooses against his own interest at any point).

Input

The chocolate brands are numerated randomly, except "Router's Ports", which is labeled 1. The input consists of

- one line containing N ($1 \leq N \leq 2 \cdot 10^5$) – the amount of chocolate brands
- N lines describing the brands, with the i-th line containing two numbers a_i and c_i ($0 \leq a_i \leq N, 0 < c_i \leq 10^9; a_1 = 0$) – ancestor and cost of the i-th chocolate.

Output

Output the cost of the chocolate that Martin gets if both he and the accountant always choose right (w.r.t. their respective goals).

Sample input and output

Input	Output
8 0 4 5 3 1 2 5 1 1 5 4 8 3 6 3 7	7

Problem B: Biased heap game

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Recall the heap game (sometimes framed as matchstick puzzle):

- there is a heap, initially consisting of n items
- two players take turns removing anywhere from 1 to k items
- the first player who can't make another move loses.

Since you had become way too good at the game after attending the game theory lecture, your old heap game buddy convinced you to modify the game in a way that he perceives to favor him. After tense negotiations you agreed on the following rule change: Instead of being allowed to take anywhere from 1 to k items per turn, he is now allowed to take anywhere from 2 to $k + 1$. In exchange, he agreed to always let you make the first move.

Of course this means you now need to adjust your strategy in order to still win as often as possible.

Input

The input consists of

- an integer T ($1 \leq T \leq 10^5$) – the number of games you play
- T lines, with the i-th containing integers n_i and k_i ($1 \leq n_i, k_i \leq 10^9$) – the parameters of the i-th game

Output

For each game output the largest amount of items you can take in the first move such that your friend cannot win the game if all your remaining moves are perfect. If no move secures the win in this way, output 0.

Sample input and output

Input	Output
3 2 1 5 3 252739 729	1 0 544

Problem C: Divide heaps

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The two kids play another heap game. Again, there initially is only one heap of n candies, but this time instead of taking away candies, they divide heaps into two heaps of different sizes. The player who cannot divide a heap anymore loses. How many of the permitted first moves secure the win for the first player (the one who makes the first move) in the optimal game of both players?

Input

The input consists of

- one line containing T ($1 \leq T \leq 10^4$) – the number of games they play
- T lines contain the descriptions of these games, with the i-th line containing n_i ($1 \leq n_i \leq 10^4$) – the amount of candies in that game.

Output

For each game output the amount of possibilities the first player has on the first turn to win in the optimal game of both players. If the first player can't win output 0.

Sample input and output

Input	**Output**
5 1 2 3 4 9	0 0 1 0 2

Problem D: Coins

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

There are N heaps of coins on the table ordered in a row. In the i-th heap there are exactly a_i coins. Additionally, it is known that $a_i \leq a_j$ if $i < j$.

Katya and Sergey play a game with these coins. At each move it is allowed to take any positive amount of coins from any heap, but the condition ($a_i \leq a_j$ if $i < j$) must remain satisfied. The player who takes the last coin, wins.

You should find who wins if no one makes a mistake and Sergey gets to make the first move.

Input

The input consists of

- one line containing N ($1 \leq N \leq 10^5$) – the number of heaps
- one line containing N numbers $a_1, a_2, ..., a_N$ ($1 \leq a_i \leq 10^9$) – the sizes of the heaps.

Output

If Sergey wins output "`Sergey`", otherwise output "`Katya`".

Sample input and output

Input	**Output**
1 10	Sergey
2 10 15	Sergey
2 10 10	Katya

Problem E: Extreme TicTacToe

Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

You are playing *Extreme TicTacToe*, a generalization of the famously unwinnable game. Here, the playing board is a 4×4 grid, in which an even number of cells (not always the same ones) are blocked from the beginning. Also, the game can't end in a draw: If all cells are used up, the player whose turn it would be to make the next move loses (this is somewhat justified: the beginning player has a small advantage and since there always is an even number of free cells in the beginning, it's always his turn when all of them are filled). Other than that, the rules are the same as for classical TicTacToe:

- the two players take turns placing their respective symbol ('X' or 'O') in one of the remaining free cells
- once a player has a horizontal, vertical or diagonal row of (at least) 3 of his symbols, he wins.

Given which cells are blocked and assuming you make the first move, find out if you would win in a perfect game of both players.

Input

The input consists of

- one line containing A ($0 \leq A \leq 10$) – the number of blocked cells
- A lines describing the blocked cells, with the i-th line containing integers r_i and c_i ($1 \leq r_i, c_i \leq 4$), denoting that the cell in row r_i and column c_i is blocked. It is guaranteed that the pairs (r_i, c_i) are pairwise distinct.

Output

Output "YES" if you would win in a perfect game of both players and "NO" otherwise.

Sample input and output

Input	Output
6 1 4 2 4 3 4 4 1 4 2 4 3	NO