A software company is currently made up of several teams with a certain number of employees in each team. A new policy is being implemented where the percentage of senior developers in the teams needs to be at or above a certain threshold. The company wants to hire new senior developers and place them into teams where needed so that this requirement is met for the company. Given the number of employees and senior developers in each team, as well as the threshold percentage, how many senior developers does the company need to recruit?

For example, let's say that there is $n = 1$ team, where *teams = [[2, 5]]*, and the threshold percentage requirement is $r = 60$. The variable *teams* denote that there are 2 senior developers in a team that has a total size of 5. Initially, the percentage of senior developers in the team is 40% (2/5), but the following occurs as senior developers are hired and added to the team:

- With the first hire, the percentage rises to 3 / 6 = 50%
- With the second hire, the percentage rises to 4 / 7 = 57.14%
- With the third hire, the percentage rises to 5 / 8 = 62.5%

At this point, the threshold requirement of 60% has been met. Therefore, the answer is 3 because that is how many senior developers the company needs to hire in accordance with the new policy.

Note: The percentage does not apply to individual teams but rather an average of all the teams. Each team has equal weight, so they must first be normalized to a common total score before calculating the percentage. For example, if senior developers make up 100% of one team and 50% of another, the total percentage is calculated as 75%.

Function Description

Complete the function *hireSeniorDevs* in the editor below.

hireSeniorDevs has the following parameters:

int *teams*[n][2]*:* a 2-dimensional array of integers where the i^{th} element contains two values, the first one denoting *senior[i]* and the second denoting *total[i]*

int *r:* the percentage of senior developers there needs to be in the teams

Returns:

int: the minimum number of senior developers the company needs to hire to meet the threshold *r*

Constraints

- $1 \le n \le 200$
- $0 \le senior < total \le 100$

- $1 \le r < 100$
- The array *teams* contain only non-negative integers.

Input Format For Custom Testing

The first line contains an integer, *n*, denoting the number of rows in the array *teams*.

The second line contains the fixed integer 2, denoting the number of columns in the array *teams*.

Each line *i* of the *n* subsequent lines (where $0 \le i < n$) contains two space-separated integers, *teams[i][0]* and *teams[i][1]*.

The last line contains an integer, *r*.

Sample Case 0

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
2     →  teams[] rows, n = 2
2     →  teams[] columns, 2
1 2   →  teams = [[1, 2], [1, 3]]
1 3
50    →  r = 50
```

Sample Output

```
1
```

Explanation

The first team is already at the threshold of 50% because there is 1 senior developer in a team of 2. By hiring an additional senior developer to the second team, the percentage would become 2 / 4 = 50%. The average percentage of all the teams is then 50%, which meets the threshold requirement. Therefore, the answer is 1 because the company needs to hire only 1 senior developer to meet the requirements.

Sample Case 1

Sample Input For Custom Testing

```
STDIN    Function
-----    --------
2     →  teams[] rows, n = 2
2     →  teams[] columns, 2
1 2   →  teams = [[1, 2], [2, 3]]
2 3
75    →  r = 50
```

Sample Output

```
3
```

Explanation

For this example, the company can hire 2 senior developers to the first team and 1 senior developer to the second team, like so:

- With 2 more senior developers in the first team (which is initially 1/2), the percentage would become 3 / 4 = 75%
- With 1 more senior developer in the second team (which is initially 2/3), the percentage would become 3 / 4 = 75%

Therefore, the answer is 3 because the company needs to hire a minimum of 3 senior developer to meet the requirements.

```
import java.io.*;
import java.math.*;
import java.security.*;
import java.text.*;
import java.util.*;
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;
class Result {


    /*
     * Complete the 'devTeam' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     *  1. 2D_INTEGER_ARRAY teams
     *  2. INTEGER r
     */
    public static int devTeam(List<List<Integer>> teams, int r) {
    // Write your code here

    }

}
public class Solution {
    public static void main(String[] args) throws IOException {
        BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(System.in));
        BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(System.getenv("OUTPUT_PATH")));

        int teamsRows = Integer.parseInt(bufferedReader.readLine().trim());
        int teamsColumns = Integer.parseInt(bufferedReader.readLine().trim());

        List<List<Integer>> teams = new ArrayList<>();

        IntStream.range(0, teamsRows).forEach(i -> {
            try {
                teams.add(
                    Stream.of(bufferedReader.readLine().replaceAll("\\s+$", "").split(" "))
                        .map(Integer::parseInt)
                        .collect(toList())
                );
            } catch (IOException ex) {
                throw new RuntimeException(ex);
            }
        });

        int r = Integer.parseInt(bufferedReader.readLine().trim());

        int result = Result.devTeam(teams, r);

        bufferedWriter.write(String.valueOf(result));
        bufferedWriter.newLine();


        bufferedReader.close();

        bufferedWriter.close();

    }

}
```