Truffa al sushi (`sushi`)

A new sushi restaurant has opened in Pordenone and, of course, Alessandro goes there immediately to review it. As soon as he enters, however, he discovers that he has been scammed, the restaurant **isn't** all-you-can-eat!

Figure 1: Alessandro ordering the first dish

Alessandro has exactly B euros in his wallet and, since he wants to feel truly satisfied at the end of his meal, he wants to spend **all** the B euros. There are N dishes on the menu, the i-th dish costs A_i euros. Alessandro can order the same dish several times, but eating too many times the same dish makes him sick. Therefore, he wants to find a way to spend B euros **minimizing** the maximum number of occurrences of the same dish.

Help Alessandro find the minimum number of occurrences of the same dish!

Implementation

You should submit a single file, with either a `.c` or `.cpp` extension.

☞ Among the attachments in this task you will find a template `sushi.c` or `sushi.cpp` with a sample implementation.

You will have to implement the following function:

C	`int sushi(int N, int B, int A[]);`
C++	`int sushi(int N, int B, vector<int> A);`

- The integer N represents the number of dishes on the menu.
- The entire B represents how many euros Alessandro wants to spend.
- The vector A, indexed from 0 to $N-1$, contains the cost of each dish.

The function `sushi` must return the minimum number of occurrences of the same dish, or -1 if it is impossible to spend exactly B euros.

Sample grader

Among this task's attachments you will find a simplified version of the grader used during the evaluation, which you can use to test your solutions locally. The sample grader reads data from `stdin`, calls the functions that you should implement and writes back on `stdout` using the following format.

The input file is formed by 2 lines, containing:

- Line 1: the integers N and B.
- Line 2: the N integers $A_0, \dots, A_{N-1}$.

The output file is formed by a single line, containing the value returned by `sushi`.

Constraints

- $1 \le N \le 10^4$
- $1 \le B \le 10^5$
- $1 \le A_i \le B$ for each i

Scoring

Your program will be tested against several test cases grouped in subtasks. In order to obtain the score of a subtask, your program needs to correctly solve all of its test cases.

- **Subtask 1** [**0 points**]: Examples.
- **Subtask 2** [**14 points**]: $N = 1$
- **Subtask 3** [**10 points**]: $N \le 5, B \le 10$
- **Subtask 4** [**15 points**]: $N \le 50, B \le 50$, also there is always a solution.
- **Subtask 5** [**10 points**]: $N \le 50, B \le 100$
- **Subtask 6** [**20 points**]: $N \le 75, B \le 300$
- **Subtask 7** [**15 points**]: $N \le 500, B \le 1000$
- **Subtask 8** [**10 points**]: $N \le 500, B \le 10000$
- **Subtask 9** [**5 points**]: $N \le 1000, B \le 100000$
- **Subtask 10** [**1 point**]: No additional limitations.

Examples

`stdin`	`stdout`

`2 15`
`1 10`
	`5`

`2 80`
`5 8`
	`8`

`3 19`
`11 6 5`
	`-1`

Explanation

In the **first example**, there are 2 dishes with costs 1 and 10 euros. Since Alessandro's budget is 15 euros, he can order the second dish at most once. If Alessandro doesn't order the second dish, he should order the first dish 15 times with a good chance that this will make him nauseous. Instead, if he orders the second dish once, he should order the first dish 5 times and he will be less likely to get nauseous. Hence, Alessandro will order each dish respectively 5 and 1 times and so the solution is $\max(1, 5) = 5$.

In the **second example**, it is optimal to order 8 times the first and 5 times the second; therefore the solution is $\max(8, 5) = 8$.

In the **third example**, it is impossible to spend exactly 19 euros.