Task: TAB
Totally Aligned Buttons

AACPP SuSe 2025 Round 7 Memory: 4MiB 2025.07.01 – 2025.07.08

Dexter the Cat, in the midst of his daily mischief marathon, was struck by a sudden flair for feline chaos. While his human was away, Dexter executed a flawless leap onto the desk, stretched out like royalty across the custom mechanical keyboard, and - **crash!** - sent it tumbling to the floor in a glorious clatter of keys.

When the owner returned, he found Dexter sitting proudly behind the keyboard, which had clearly been knocked over and then paw-sibly reassembled. Dexter was purring with smug satisfaction, as if to say, "You're welcome." The keyboard, however, looked nothing like it did before. It seems Dexter remembered only a few things:

- There were some special keys in fixed spots (like the big spacebar he loves to nap on),
- And the rest of the layout was *probably* made up of **repeatable columns of keys** (perfect for pressing randomly while loafing).

In his infinite cat-fidence, Dexter figured that if he could match the **total number of keys** in two different layouts - each with its own number of special keys and repeatable column size - then maybe, just maybe, his human would believe he'd actually helped and forgive the whole kerfuffle.

So now, going a bit back in time before the great keyboard cat-astrophe, your task is to help this whiskered rascal prove his purr-sumed genius and figure out wherther it's paw-sible to build both layouts using the same total number of keys, by repeating the columns as many times as needed.

Input

The first and only line of input contains five integers s_1, c_1, s_2, c_2, and n. Here, s_1 and c_1 are the numbers of special keys and the number of keys per column in the first design, s_2 and c_2 correspondingly in the second design, and n is the minimal acceptable number of keys.

Output

Print a single integer k, equal to the **minimum number of keys** that can be used to implement both designs. If there is no such k, output `IMPAWSIBLE`. Note that if there are enough special keys, it might be possible to implement a design using zero columns.

Example

For the input:

```
10 7 8 5 40
```

the correct output is:

```
73
```

For the input:

```
1 3 2 4 15
```

the correct output is:

```
22
```

For the second example, without the minimal key requirement, the answer would be `10`, as the first layout can be implemented with 1 special key and 3 columns, and the second layout with 2 special keys and 2 columns, both giving a total of 10 keys. However, since the minimal number of keys is 15, we need to repeat the columns more to reach that number, resulting in a total of 22 keys ($1 + 3 * 7, 2 + 4 * 5$).

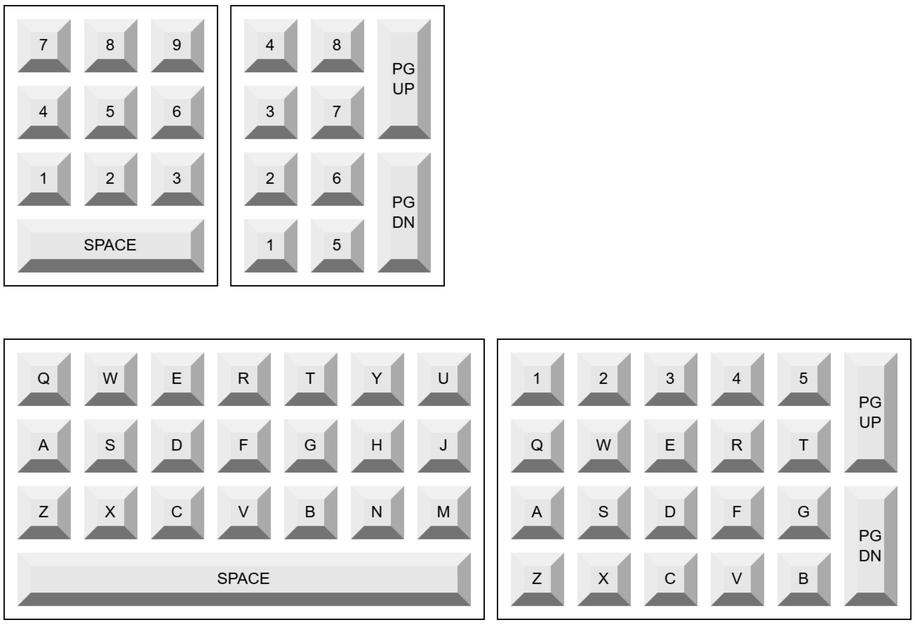

Figure 1: Second sample visualization.

Additional examples

The following initial tests are also available:

- `0c` – $n = 0$, an impawsible task;
- `0d` – $n = 967$, the minimum number of keys matches the number of special keys in the bigger design;
- `0e` – $n = 5337$, large impawsible task;

Limits

Your solution will be evaluated on a number of hidden test cases divided into groups. Points for a group are awarded if and only if the submission returns the correct answer for each of the tests in the group within the allotted time limit. These groups are organised into subtasks with the following limits and points awarded.

For all tests, $0 \le n \le k \le 10^{18}$ if k exists.

Subtask	**Limits**	**Points**
1.	$1 \le n \le 100$, $0 \le s_1, c_1, s_2, c_2 \le 500$	2
2.	$1 \le n \le 1\,000$, $0 \le s_1, c_1, s_2, c_2 \le 500\,000$	3
3.	$1 \le n \le 10^{15}$, $0 \le s_1, c_1, s_2, c_2 \le 10^{15}$	5