

## B. Memory Manager

time limit per test: 1 second  
 memory limit per test: 64 megabytes

There is little time left before the release of the first national operating system BerIOS. Some of its components are not finished yet — the memory manager is among them. According to the developers' plan, in the first release the memory manager will be very simple and rectilinear. It will support three operations:

- `alloc n` — to allocate  $n$  bytes of the memory and return the allocated block's identifier  $x$ ;
- `erase x` — to erase the block with the identifier  $x$ ;
- `defragment` — to defragment the free memory, bringing all the blocks as close to the beginning of the memory as possible and preserving their respective order;

The memory model in this case is very simple. It is a sequence of  $m$  bytes, numbered for convenience from the first to the  $m$ -th.

The first operation `alloc n` takes as the only parameter the size of the memory block that is to be allocated. While processing this operation, a free block of  $n$  successive bytes is being allocated in the memory. If the amount of such blocks is more than one, the block closest to the beginning of the memory (i.e. to the first byte) is preferred. All these bytes are marked as not free, and the memory manager returns a 32-bit integer numerical token that is the identifier of this block. If it is impossible to allocate a free block of this size, the function returns `NULL`.

The second operation `erase x` takes as its parameter the identifier of some block. This operation frees the system memory, marking the bytes of this block as free for further use. In the case when this identifier does not point to the previously allocated block, which has not been erased yet, the function returns `ILLEGAL_ERASE_ARGUMENT`.

The last operation `defragment` does not have any arguments and simply brings the occupied memory sections closer to the beginning of the memory without changing their respective order.

In the current implementation you are to use successive integers, starting with 1, as identifiers. Each successful `alloc` operation procession should return following number. Unsuccessful `alloc` operations do not affect numeration.

You are to write the implementation of the memory manager. You should output the returned value for each `alloc` command. You should also output `ILLEGAL_ERASE_ARGUMENT` for all the failed `erase` commands.

### Input

The first line of the input data contains two positive integers  $t$  and  $m$  ( $1 \leq t \leq 100; 1 \leq m \leq 100$ ), where  $t$  — the amount of operations given to the memory manager for processing, and  $m$  — the available memory size in bytes. Then there follow  $t$  lines where the operations themselves are given. The first operation is `alloc n` ( $1 \leq n \leq 100$ ), where  $n$  is an integer. The second one is `erase x`, where  $x$  is an arbitrary 32-bit integer numerical token. The third operation is `defragment`.

### Output

Output the sequence of lines. Each line should contain either the result of `alloc` operation procession, or `ILLEGAL_ERASE_ARGUMENT` as a result of failed `erase` operation procession. Output lines should go in the same order in which the operations are processed. Successful procession of `alloc` operation should return integers, starting with 1, as the identifiers of the allocated blocks.

#### → Attention

The package for this problem was not updated by the problem writer or Codeforces administration after we've upgraded the judging servers. To adjust the time limit constraint, a solution execution time will be multiplied by 2. For example, if your solution works for 400 ms on judging servers, then the value 800 ms will be displayed and used to determine the verdict.

#### Codeforces Beta Round 7

Finished

Practice



#### → Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

#### → Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

#### → Submit?

Language: GNU G++23 14.2 (64 bit, ms)

Choose file:  No file chosen

Submit

#### → Last submissions

Submission	Time	Verdict
<a href="#">324370927</a>	Jun/14/2025 15:44	Accepted

## Examples

### input

[Copy](#)

```
6 10
alloc 5
alloc 3
erase 1
alloc 6
defragment
alloc 6
```

### output

[Copy](#)

```
1
2
NULL
3
```

## → Problem tags

[implementation](#) [\\*1600](#)[No tag edit access](#)

## → Contest materials

- Codeforces Beta Round #7 [×](#)

[Codeforces](#) (c) Copyright 2010-2025 Mike Mirzayanov

The only programming contests Web 2.0 platform

Server time: Jun/14/2025 19:45:04<sup>UTC+7</sup> (k1).

Desktop version, switch to [mobile version](#).

[Privacy Policy](#) | [Terms and Conditions](#)

Supported by



**ITMO**