A. VM scheduling

Limits: 4 sec., 1024 MiB

A typical cloud-based virtual machine service provides you a functionality to work with virtual machines (VM). Let's consider one of those services.

Each VM is characterized by two parameters: the amount of RAM and the number of CPU cores required for running this particular VM. Each VM runs on a single server.

Each server consists of two nodes: A and B. Each server node has exactly m_{server} GB of RAM and c_{server} CPU cores. Some types of VMs are to be run entierly on one server node while others have to be allocated on two nodes of a single server. In the latter case both of the nodes should allocate half of the memory and CPU resources needed for this VM. A server can run multiple VMs as long as the total memory and the number of CPU cores of the VMs don't exceed capacities of server nodes.

Your task is to implement a scheduler for such cloud based VM-service.

You are to process n requests of two types:

1. VM create request: allocate a new VM on a server.
2. VM delete request: stop one of the previously created VMs.

For each create request you have to decide which server and nodes to run the VM on. After a VM is associated with a server it runs on it occupying certain amount of resources until the VM is deleted. After a VM is deleted the resources are released and can be used by other VMs.

Your task is to minimize the total number of servers used.

Input

The first line of the input contains three number n m_{server} c_{server} — the total number of requests to be processed, amount of RAM and CPU cores on a server node. The i-th of the next n lines has one of the following two formats:

- `0` m_i c_i t_i — create request for a VM with m_i GB of RAM and c_i CPU cores which has to be allocated on t_i server nodes. This VM has ID i (1-based).
- `1` id_i — delete a VM with ID id_i.

It is guaranteed that for each delete request the VM with specified ID is still running.

Output

First line of the output should contain one integer k — the total number of servers used by your scheduler. Servers are numbered from 1 to k.

For each create request first you have to print one integer — the number of the server which the current VM should be run on. In case the VM should be run only on one node, you should print either `A` or `B` depending on the server node this VM should be run on. The server number should be valid and corresponding server nodes should have the necessary amount of resources to run the VM.

For each delete request you shouldn't print anything.

For each test case you are allowed to use at most n servers.

Constraints

$1 \le n \le 5 \cdot 10^5$,
$1 \le m_{server}, c_{server} \le 500$,
$1 \le c_i \le c_{server}$, $1 \le m_i \le m_{server}$ for $t_i = 1$,
$1 \le c_i \le 2 \cdot c_{server}$, $1 \le m_i \le 2 \cdot m_{server}$ for $t_i = 2$,

$1 \le t_i \le 2$,
For each create request with $t_i = 2$ m_i and c_i are even,
$1 \le id_i < i$

Samples

Input (*stdin*)	Output (*stdout*)
8 16 32 0 8 16 1 0 2 4 1 0 8 16 2 1 1 0 8 16 1 1 5 1 3 0 8 16 1	2 1 A 1 A 1 2 A 2 B

Notes

You can print any valid answer your solution can find, not necessarily the most optimal one.
In the sample each server node has 16 GB of RAM and 32 CPU cores. There are 8 requests:

1. `0 8 16 1` — create a VM with 8 GB of RAM and 16 CPU cores that should be allocated on 1 server node.

   Output:

   `1 A` — allocate this VM on server 1 node A

2. `0 2 4 1` — create a VM with 2 GB of RAM and 4 cores that should be allocated on 1 server node.

   Output:

   `1 A` — allocate this VM on server 1 node A

3. `0 8 16 2` — create a VM with 8 GB of RAM and 16 cores that should be allocated on 2 server nodes. On both nodes 4 GB of RAM and 8 cores will be used.

   Output:

   `1` — allocate this VM on both server 1 nodes.

4. `1 1` — delete the VM created on the first request.

5. `0 8 16 1` — create a VM with 8 GB of RAM and 16 cores that should be allocated on 1 server node.

   Output:

   `2 A` — allocate this VM on server 2 node A.

6. `1 5` — delete the VM created on the fifths request.

7. `1 3` — delete the VM created on the third request.

8. `0 8 16 1` — create a VM with 8 GB of RAM and 16 cores that should be allocated on 1 server node.

   Output:

   `2 B` — allocate this VM on server 2 node B.

Scoring

Your raw score for a test case is k — the total number servers used for this test case.

The normalized score for a test case is

$$\left\lfloor \frac{OPT \cdot 10^7}{RAW} \right\rfloor,$$

where RAW is your raw score and

$$OPT = \max\left(\left\lceil \frac{MAX_{memory}}{2 \cdot m_{server}} \right\rceil, \left\lceil \frac{MAX_{cpu}}{2 \cdot c_{server}} \right\rceil\right).$$

Here MAX_{memory} is the maximum total memory of all running VMs during the whole process. Similarly, MAX_{cpu} is the maximum total cpu of all running VMs over all the requests.

Note that RAW is always greater than or equal to OPT which means your normalized score for a test case does not exceed 10^7.

In case if the output for a test case is invalid (no output, time limit exceeded, run time crash, wrong output format, using more than n servers, allocating a VM on wrong number of server nodes, allocating VMs on nodes without enough free resources etc.), the normalized score is 0.

Finally, your overall score is the sum of your normalized score over all test cases.

Quick start

Check the sample solution which tries to allocate each VM on the last server. In case it was not possible the solution adds a new server. The source code is available for all contest programming languages:

- C++
- Python
- Java
- C#
- Pascal
- F#

Submissions

- The execution time limit is 4 seconds per test case and the memory limit is 1024 mibibytes.
- The code size limit is 64 kibibytes.
- The compilation time limit is 1 minute.
- There are 50 provisional test cases. Your submissions will be evaluated on provisional set during the submission phase.
- You can submit your code once per 30 minutes and you will get a feedback with your normalized score for each of the provisional tests.
- There will be 500 test cases in the final testing after the submission phase is over. The final results will be announced during two weeks.

Tests

All test cases including provisional and final sets are generated based on the data from real data centers that manage a pool of virtual machines. Each test case contains requests for some specific period of time. As the data from the data center is confidential the tests cases (provisional and final) are not publicly available.