#### LRu Cache

#### **Description**

Implement a LRu Cache with capacity N, you need to follow the constraints of LRu.

There are M operations including two type:

- **get key**: Print the value of the key if the key exists, otherwise print -1.
- **put key value**: It the key exists, update the key value. Otherwise, add the key-value pair to the cache. If the number of keys exceeds the capacity, you should evict the least recently used key.

In addition, each operation will cost 1 time unit. Before each operation, you need to check that if any key in the cache have not been used for more than T time units. You must evict them all before each operation.

### **Input Format**

The first line contains 3 integers, N , M and T.

Then, the next M lines represent for M operations.

For each line, first input an integer **op**, **op** = 1 means that it is a get operation, **op** = 2 means that it is a put operation.

if op = 1, then input an integer k, represents the key.

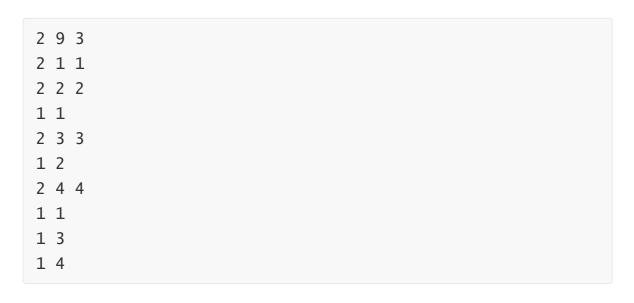
if  $\mathbf{op} = 2$ , then input two integers k, v, represent the key and the value.

#### **Output Format**

The output contains K+1 lines. For each  $\mbox{\bf get}$  operation, print the value of corresponding key.

Also, you should output the all the values of the keys that end up in the cache, in the order of update time.

# **Sample Input**



## **Sample Output**

```
1
-1
-1
-1
4
4
```

```
For 100\% testcases, 1\leq N, T\leq 2\times 10^4, 1\leq M\leq 4\times 10^5, 1\leq key\leq 3\times 10^4.
```