Least Recently Used Cache

Assume that you have to design a new data structure the Least Recently Used Cache (LRU). LRU is a common caching strategy. It defines the policy to remove elements from the cache to make room for new elements when the cache is full, meaning it discards the least recently used items first. For example, we cache elements [1, 2, 3, 4] and now need to cache another element “5”. In LRU cache, we evict the least recently used element (in this case “1”) and the cache will be [2, 3, 4, 5]. Now, “2” is the next element to be removed if a new element needs to be cached, but when “2” is accessed again, then “3” becomes the next element to be removed from the cache and the cache will be [3, 4, 5, 2].

Write a **linked list based program** that should take as input, number of operations P and the capacity of the cahce, C. Each operation can be one of the below.

- **1 K V** -Set the value (V) of the key (K) if the key exists. Otherwise, add the key-value pair to the cache. For any error while insertion, print "Error".
- **2 k** -Print the value of the key (K) from the cache if the key exists, otherwise print "Not Found".

Grading will be based on the *correctness* and the *performance*, so do your best to wrire a clean and optimal code.

Input Format

The first line of input contains two integers, P and C. The next P lines each contains an operation like the mentioned above.

Constraints

- 1 <= P <= 10^6
- 0 <= C <= 10^6
- 1 <= E <= 10^9

Output Format

Print the value of the key in the cache when requested.

Sample Input 0

```
5 3
1 1 100
1 2 200
1 3 300
1 4 400
2 3
```

Sample Output 0

```
300
```

Sample Input 1

```
7 5
```

```
1 1 100
2 1
1 2 200
1 3 300
2 2
1 4 400
2 4
```

Sample Output 1

```
100
200
400
```