Storage

Mootopia is a magical city in the distant land of Hogsmeade. Every year, hundreds and thousands of tourists visit Mootopia, known for its many unique souvenir offerings. The souvenirs sold in Mootopia are one-of-a-kind and could not be found elsewhere in the world. Visiting Mootopia is considered a must for every travel-savvy individual.

At the moment, Nathan is travelling in Mootopia. As a tourist, Nathan wants to buy unique Mootopia souvenirs for his friends and families. Every souvenir that Nathan buys is different and has a unique name. Each souvenir also has a value associated with it. Nathan definitely wants to buy as much as he possibly could. However, Nathan can only carry at most S souvenirs with him at any given time.

Fortunately, he has a magical device which can automatically send souvenirs from Mootopia to his magical storage located somewhere secure. His storage is organized into N boxes, numbered from 1 to N, with each box being able to store at most K items.

His magical device can manage his inventory well. Therefore, while in Mootopia, Nathan can do all these things without having to worry about a single thing:

1. Purchase a new item with a specific value.
2. Deposit an item to his storage.
3. Withdraw an item from his storage.
4. Find the location of a specific item.
5. Find the most valuable item that he owns.

To use this device, it needs to be programmed first. Therefore, Nathan needs you, the best programmer in the world, to write a program that is able to process each of the above-mentioned queries.

Good luck!

Input

The first line of input consists of four integers N (10 <= N <= 50), S (1 <= S <= 10), K (5 <= K <= 30), and Q (5 <= Q <= 200), each separated by a single space, representing the number of boxes in Nathan's storage, the number of items that Nathan can carry, the number of items that can fit in one box, and the number of queries being asked.

The next Q lines will contain a single query each with the following format:

Query Type Input Format: `<QUERY_TYPE> <APPROPRIATE_PARAMETERS>`

1. `purchase ITEM_NAME VALUE`
   Buy the item `ITEM_NAME` with the value `VALUE`. The value is guaranteed to be a positive integer at most 10000 and the item name is guaranteed to be unique (i.e. it has not been purchased before). If Nathan is currently carrying less than S items with him, he will carry `ITEM_NAME` with him. Otherwise, he sends the item to the lowest-numbered box in his storage that does not already have K items. If the item is with him after the purchase, print "item `ITEM_NAME` is now being held". Otherwise, print "item `ITEM_NAME` has been deposited to box `BOX_NUMBER`".

   It is guaranteed that all item names consist of lowercase English letters only.

2. `deposit ITEM_NAME`
   Deposit the item `ITEM_NAME` to storage. Specifically, put the item to the lowest-numbered box that is not full. If the item is already in storage, print "item `ITEM_NAME` is already in storage". If Nathan has not bought the item `ITEM_NAME`, print "item `ITEM_NAME` does not exist". Otherwise, print "item `ITEM_NAME` has been deposited to box `BOX_NUMBER`".

3. `withdraw ITEM_NAME`
   Withdraw the item `ITEM_NAME` from storage so that Nathan holds the item. If Nathan is currently holding the item, print "item `ITEM_NAME` is already being held". If Nathan has not bought the item `ITEM_NAME`, print "item `ITEM_NAME` does not exist". If the item is in storage but Nathan cannot hold any more items (i.e. he is currently holding `S` items), print "cannot hold any more items". Otherwise, print "item `ITEM_NAME` has been withdrawn".

4. `location ITEM_NAME`
   Print the location of the item `ITEM_NAME`. If Nathan has not bought the item, print "item `ITEM_NAME` does not exist". If he is currently holding the item, print "item `ITEM_NAME` is being held". Otherwise, print "item `ITEM_NAME` is in box `BOX_NUMBER`".

5. `valuable`
   Print the name of the most valuable item that Nathan owns. If there is more than one item with the same highest value, print the one with the lexicographically-smallest name.

It is guaranteed the storage can always hold all the items being bought, i.e. even though the storage is "limited", it is guaranteed that Nathan would not buy items more than the storage can handle.

Output

Print the result of the query as described in the input format above. The last line of the output should contain a newline character. In the sample input below, the first line is left empty for better clarity of the sample. No blank line is to be printed in the actual output.

Sample Input	Sample Output
2 2 5 11	
withdraw candy	item candy does not exist
purchase candy 100	item candy is now being held
purchase chocolate 200	item chocolate is now being held
purchase iphone 500	item iphone has been deposited to box 1
location candy	item candy is being held
valuable	iphone
deposit iphone	item iphone is already in storage
withdraw iphone	cannot hold any more items
deposit candy	item candy has been deposited to box 1
purchase air 500	item air is now being held
valuable	air

Explanation

`2 2 5 11`	Nathan has 2 boxes, can carry at most 2 items with him, each box can contain at most 5 items, and there are 11 queries.
`withdraw candy`	Nathan has not bought any items. Hence he does not have the item “candy”.
`purchase candy 100`	The item “candy” with value 100 is purchased and is now held.
`purchase chocolate 200`	Nathan bought “chocolate” with value 200. The item is now held since Nathan can hold 1 more item.
`purchase iphone 500`	Nathan bought “iphone” with value 500. However, he is currently carrying 2 items already, so he needs to send “iphone” to box 1, the smallest-numbered available box.
`location candy`	“candy” is currently with Nathan.
`valuable`	Item “iphone” is currently the most valuable item.
`deposit iphone`	“iphone” is already in storage, specifically box 1.
`withdraw iphone`	Nathan cannot withdraw “iphone” since he is already carrying 2 items with him, the most for this sample test case.
`deposit candy`	Nathan deposits candy to box 1 and is now only holding the item “chocolate”.
`purchase air 500`	Nathan buys the item “air” (really?) with value 500. It is now held since Nathan is currently carrying only 1 item, which is “chocolate”.
`valuable`	The item “air” is now the most valuable item with value 500, compared to “candy” (100), “chocolate” (200), and “iphone” (500). “air” and “iphone” both have the same value, but “air” is lexicographically smaller than “iphone”.

Skeleton

You are given the file **Storage.java**[1]. You should see the following contents when you open the file, otherwise you are in the wrong directory.

```
/**
 * Name         :
 * Matric No.   :
 * Plab Acct.   :
 */

public class Storage {

    public void run() {
        // treat this as your "main" method
    }

    public static void main(String[] args) {
        Storage myStorageSystem = new Storage();
        myStorageSystem.run();
    }

}

class Box {
    // define appropriate attributes, constructor, and methods
}

class Item {
    // define appropriate attributes, constructor, and methods
}
```

Notes:

1. You should develop your program in the subdirectory **ex1** and use the skeleton java file provided. You should not create a new file or rename the file provided.
2. You only need to modify the skeleton file, that is **you do not need to create a new file for each class.** All code should be **inside the file given to you** in the ex1 directory.
3. If your algorithm is different from the given skeleton, you are free to write a solution according to your own algorithm. You are free to define your own classes besides the ones given in the skeleton file.
4. You **must (and need to)** use OOP for this sit-in lab.
5. You are free to define your own methods.
6. Please be reminded that the marking scheme is:

   Input : 10%
   Output : 10%
   Correctness : 50%
   Programming Style : 30%, which consists of:
   - Meaningful comments (pre- and post- conditions, comments inside the code): 10%
   - Modularity (incremental programming, proper modifiers [public / private]): 10%
   - Proper Indentation: 5%
   - Meaningful Identifiers (for both method and variable names): 5%

   Compilation Error : Deduction of **50% of the total marks obtained**.

[1] The class "Storage" here represents the main / controller class. Do not get confused by its naming. It is not supposed to only contain the items in the storage system (boxes), but to represent the program as a whole.