

## 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, your friend Evan is travelling in Mootopia. As a tourist, Evan wants to buy unique Mootopia souvenirs for his friends and families. Every souvenir that Evan buys is different and has a unique name. Each souvenir also has a value associated with it. Evan definitely wants to buy as much as he possibly could. However, Evan 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, Evan 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, Evan 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** ( $10 \leq N \leq 50$ ), **S** ( $1 \leq S \leq 10$ ), **K** ( $5 \leq K \leq 30$ ), and **Q** ( $5 \leq Q \leq 200$ ), each separated by a single space, representing the number of boxes in Evan's storage, the number of items that Evan 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 Evan 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 Evan 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 Evan holds the item. If Evan is currently holding the item, print “item **ITEM\_NAME** is already being held”. If Evan has not bought the item **ITEM\_NAME**, print “item **ITEM\_NAME** does not exist”. If the item is in storage but Evan 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 Evan 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 Evan 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 Evan 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

```

2 2 5 11
withdraw candy
purchase candy 100
purchase chocolate 200
purchase iphone 500
location candy
valuable
deposit iphone
withdraw iphone
deposit candy
purchase air 500
valuable

```

#### Sample Output

```

item candy does not exist
item candy is now being held
item chocolate is now being held
item iphone has been deposited to box 1
item candy is being held
iphone
item iphone is already in storage
cannot hold any more items
item candy has been deposited to box 1
item air is now being held
air

```

**Explanation**

2 2 5 11

Evan 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

Evan 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

Evan bought “chocolate” with value 200. The item is now held since Evan can hold 1 more item.

purchase iphone 500

Evan 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 Evan.

valuable

Item “iphone” is currently the most valuable item.

deposit iphone

“iphone” is already in storage, specifically box 1.

withdraw iphone

Evan cannot withdraw “iphone” since he is already carrying 2 items with him, the most for this sample test case.

deposit candy

Evan deposits candy to box 1 and is now only holding the item “chocolate”.

purchase air 500

Evan buys the item “air” (really?) with value 500. It is now held since Evan 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**<sup>1</sup>.

<sup>1</sup> 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.