1. JavaScript: Inventory List

In this challenge, the task is to implement a function *inventoryList* such that*:*

- it maintains the collection of all item names existing in an inventory, where each item is uniquely identified by a name.
- returns a new object, with three methods:
  - add(name) - The string *name* parameter is passed, and it is added to the collection. It is guaranteed that at any time, if an item is in the collection, then no other item with the same name will be added to the collection.
  - remove(name) - The string *name* parameter is passed, and this item is removed from the collection if it exists. If it does not exist, nothing happens.
  - getList() - This returns an array of names of items added so far. The names are returned in the order the corresponding items were added.

Your implementation of the function will be tested by a stubbed code on several input files. Each input file contains parameters for the functions call. The functions will be called with those parameters, and the result of their executions will be printed to the standard output by the provided code. The stubbed code joins the strings returned by *getList* function by a comma and prints to the standard output. If *getList* returns an empty array, the stubbed code prints 'No Items'.

Constraints:

- The size of the collection will not exceed 10 at any point.
- All names passed to add(name) and remove(name) are non-empty.

▼ Input Format For Custom Testing

In the first line, there is an integer, *n*, denoting the number of operations to be performed. Each line *i* of the *n* subsequent lines (where $0 \leq i < n$) contains space-separated strings such that the first of them is a function name, and the remaining ones, if any, are parameters for that function.

▼ Sample Case 0

Sample Input For Custom Testing

```
5
add Shirt
add Trouser
getList
remove Shirt
getList
```

Sample Output

```
Shirt,Trouser
```

```
Trouser
```

Explanation

Items 'Shirt' and 'Trouser' are added by the *add* function. Then, *getList* is called and the result is printed. Item 'Shirt' is removed by calling the *remove* function. Finally, *getList* is called and the result is printed.

▼ Sample Case 1

Sample Input For Custom Testing

```
3
add Shirt
remove Trouser
getList
```

Sample Output

```
Shirt
```

Explanation

Item 'Shirt' is added by the *add* function. Then, *remove* is called with 'Trouser', but since it does not exist, nothing happens. Finally, *getList* is called and the result is printed.