Create 3 different string collections - **AddCollection**, **AddRemoveCollection** and **MyList**.

The **AddCollection** should have:

* Only a single method **Add** which adds an item to the **end** of the collection.

The **AddRemoveCollection** should have:

* An **Add** method- which adds an item to the **start** of the collection.
* A **Remove** method, which removes the **last** item in the collection.

The **MyList** collection should have:

* An **Add** method, which adds an item to the **start** of the collection.
* A **Remove** method, which removes the **first** element in the collection.
* A **Used** property, which displays the number of elements currently in the collection.

Create **interfaces**, which define the functionality of the collection, think about how to **model the relations** between interfaces to **reuse code**. Add an extra bit of functionality to the methods in the custom collections, **Add()** methods should return the index in which the item was added, **Remove** methods should **return the item** that was **removed**.

Your task is to **create** a **single copy of your collections**, after which on the **first input line** you will **receive** a **random** **number of strings** in a single line **separated by spaces** - the **elements** you must **add to each of your collections**. For each of your collections **write a single line** in the output that holds the results of all **Add operations** separated by spaces (check the examples to better understand the format). On the **second input line**, you will receive a **single** **number** - the **amount** of **Remove operations** you have to call on each collection. In the same manner, as with the **Add** operations for each collection (except the **AddCollection**), print a line with the results of each **Remove** operation separated by spaces.

### Input

The input comes from the console. It will hold two lines:

* The first line will contain a random number of strings separated by spaces - the elements you have to **Add** to each of your collections.
* The second line will contain a single number - the amount of **Remove** operations.

### Output

The output will consist of 5 lines:

* The first line contains the results of all **Add** operations on the **AddCollection** separated by spaces.
* The second line contains the results of all **Add** operations on the **AddRemoveCollection** separated by spaces.
* The third line contains the result of all **Add** operations on the **MyList** collection separated by spaces.
* The fourth line contains the result of all **Remove** operations on the **AddRemoveCollection** separated by spaces.
* The fifth line contains the result of all **Remove** operations on the **MyList** collection separated by spaces.

### Constraints

* All collections should have a **length of 100.**
* There will never be **more than 100** add operations.
* The number of removed operations will never be more than the number of added operations.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| popcorn cola donuts  3 | 0 1 2  0 0 0  0 0 0  popcorn cola donuts  donuts cola popcorn |
| one two three four five six seven  4 | 0 1 2 3 4 5 6  0 0 0 0 0 0 0  0 0 0 0 0 0 0  one two three four  seven six five four |

### Hint

Create an interface hierarchy representing the collections. You can use a List as the underlying collection and implement the methods using the List’s Add, Remove and Insert methods.