Lists

When we talk about storing multiple values in a container-like data structure, the first thing that comes to mind is a *list*.

You can initialize a list as:

```
>>> arr = list()
or simply
>>> arr = []
```

or with a few elements as:

```
>>> arr = [1,2,3]
```

Elements can be accessed easily similar to most programming languages:

```
>>> print arr[0]
1
>>> print arr[0] + arr[1] + arr[2]
6
```

Lists in Python are very versatile. You can add almost anything in a Python list.

In Python, you can create a list of any objects: strings, integers, or even lists. You can even add multiple types in a single list!

Let's look at some of the methods you can use on list.

1.) *append(x)*
Adds a single element $'x'$ to the end of a list.

```
>>> arr.append(9)
>>> print arr
[1, 2, 3, 9]
```

2.) *extend(L)*
Merges another list $'L'$ to the end.

```
>>> arr.extend([10,11])
>>> print arr
[1, 2, 3, 9, 10, 11]
```

3.) *insert(i,x)*
Inserts element $'x'$ at position $'i'$.

```
>>> arr.insert(3,7)
>>> print arr
[1, 2, 3, 7, 9, 10, 11]
```

4.) *remove(x)*
Removes the first occurrence of element $'x'$.

```
>>> arr.remove(10)
>>> arr
[1, 2, 3, 7, 9, 11]
```

5.) *pop()*
Removes the last element of a list. If an argument is passed, that index item is popped out.

```
>>> temp = arr.pop()
>>> print temp
11
```

6.) *index(x)*
Returns the first index of a value in the list. Throws an error if it's not found.

```
>>> temp = arr.index(3)
>>> print temp
2
```

7.) *count(x)*
Counts the number of occurrences of an element $'x'$.

```
>>> temp = arr.count(1)
>>> print temp
1
```

8.) *sort()*
Sorts the list.

```
>>> arr.sort()
>>> print arr
[1, 2, 3, 7, 9]
```

9.) *reverse()*
Reverses the list.

```
>>> arr.reverse()
>>> print arr
[9, 7, 3, 2, 1]
```

Task
Initialize your list (`L = []`) and follow the N commands given over N lines.

Each command will be 1 of the 8 commands given above. The *extend(L)* method will not be used. Each command will have its own value(s) separated by a space.

Input Format

The first line contains an integer, N (the number of commands).
The N subsequent lines each contain one of the 8 commands described above.

Sample Input

```
12
insert 0 5
insert 1 10
insert 0 6
```

```
print
remove 6
append 9
append 1
sort
print
pop
reverse
print
```

Sample Output

```
[6, 5, 10]
[1, 5, 9, 10]
[9, 5, 1]
```