# Exercise Session n. 3

**Algorithms and Data Structures**

---

Clone the following repository: **https://github.com/jindosanda/algo_tests** or run `git pull` if you already clone it, and place your programs inside the exercises folder under the appropriate session folder (session_3/exercises)
Read the README files to run the tests!

We will work with arrays. Remember that you are not allowed to use the `del` operation (e.g., `del A[i]`) nor any list-splicing operation (e.g., `A[i:i+1] = []`), nor other operation that does not have a constant-time complexity (e.g., `A.remove(x)` or `x in A`). In practice, the only things you can do with an array is to use its elements as variables (e.g., `A[i] = x`) or add a single element at the end (e.g., `A.append(x)`) or remove a single element at the end (`A.pop()` specifically without parameters).

## Removing a Given Element from an Array (array_rem.py)

Write a function `array_remove_pos(A,i)` that, given an array $A$ and a position $i$, removes the element `A[i]`. The complexity of `array_remove(A,x)` must be $O(1)$.

### Examples

```
>>> A = [7, 5, 2, 3, 5, 7, 3, 4, 2, 1]
>>> len(A)
10
>>> array_remove_pos(A,7)
>>> len(A)
9
>>> 4 in A
False
```

## Removing Elements from an Array (array_rem.py)

Write a function `array_remove_value(A,x)` that, given an array $A$ and a value $x$, changes $A$ so that it does not contain any value equal to $x$. The complexity of `array_remove(A,x)` must be $O(n)$, and `array_remove(A,x)` must operate on

array $A$ *in-place,* so it may only allocate a constant amount of memory.

### Examples

```
>>> A = [7, 5, 2, 3, 5, 7, 3, 4, 2, 1]
>>> len(A)
10
>>> array_remove_value(A,7)
>>> len(A)
8
>>> 7 in A
False
```

---

# Removing Elements from an Array While Maintaining Order (array_rem.py)

If you have not already done that with `array_remove_value(A,x)`, write a function `array_remove_value_stable(A,x)` that, given an array $A$ and a value $x$, changes $A$ so that it does not contain any value equal to $x$, and leaves the remaining elements in the original order within $A$. The complexity of `array_remove_value_stable(A,x)` must be $O(n)$. Also, `array_remove_value_stable(A,x)` must operate on array $A$ *in-place.*

### Examples

```
>>> A = [7, 5, 2, 3, 5, 7, 3, 4, 2, 1]
>>> len(A)
10
>>> array_remove_value_stable(A,7)
>>> len(A)
8
>>> A
[5, 2, 3, 5, 3, 4, 2, 1]
```