PIC 16: Homework 1 (due 4/9 at 10pm)

In every question, you are given an empty function with a `docstring` and you have to define it correctly. In every question, you are given simple test cases, but should be careful to think of more complicated ones in case these break your code!

How should your answers be submitted?

- Download `hw1.py`. Do NOT change the filename.
- Replace `mjandr` by your name.
- Delete the `pass` statements and type your answers. Do NOT change the function names.
- Do NOT include your test cases.

  This will hinder the grading process and might result in `0` points.

  I suggest solving the homework in one `.py` file, and then carefully transferring your work to `hw1.py`. Confirm you didn't mess up by running your test cases *at the bottom*, and then make sure to *delete them after this*.
- Submit `hw1.py` to CCLE.

Your homework will be graded by running fairly similar test cases to the ones provided.
I will carefully check things like `L is unaffected by the function call` and `None is returned`.

1. (a)
```
def compare_val_w_idx(L):
    """Compares each value of a list of numbers with its index...

    L is unaffected by the function call.
    The function returns a list M of the same length as L.

    M[i] is  1 when L[i] is bigger than i,
    M[i] is -1 when L[i] is  less  than i,
    M[i] is  0 when L[i] is  equal  to  i.
    """

    pass
```

Test:

```
L = [4,5,6,3,4,5,2,3,4]
print(compare_val_w_idx(L))
```

should print `[1, 1, 1, 0, 0, 0, -1, -1, -1]`.

(b)
```
def compare_val_w_idx_no_obj(L):
    """Compares each value of a list of numbers with its index...

    No new objects (except ints) are created during the function call!!
    None is returned, but L is mutated by the function call.

    For any list of numbers L,
    the following code results in True being printed.

    M = compare_val_w_idx(L)
    compare_val_w_idx_no_obj(L)
    print(L == M)

    So L is mutated to the return value of compare_val_w_idx(L).
    """

    pass
```

Test:

```
L = [4,5,6,3,4,5,2,3,4]
compare_val_w_idx_no_obj(L)
print(L)
```

should print `[1, 1, 1, 0, 0, 0, -1, -1, -1]`.

"No new objects" means that you should not be creating any new lists. For example,

```
def compare_val_w_idx_no_obj(L):
    M = compare_val_w_idx(L)
    L.clear()
    L.extend(M)
```

is NOT an okay solution.

(For the pedantic, I'm not counting `range(len(L))` as an object even though it is. You're fine to use this.)

2. (a)
```
def keep_first_occurrences(L):
    """Deletes duplicates in a list of numbers.

    L is unaffected by the function call.
    The function returns a list M.

    M consists of the same numbers as in L in the same order.
    However, all but the first occurrence of each number has been deleted.
    """

    pass
```

Test:

```
L = [1,2,3,1,4,1,5,2,6,6,6,5,3,2,2,2,2,3,1,1,1,1,1,4,3,1,2,1]
print(keep_first_occurrences(L))
```

should print `[1, 2, 3, 4, 5, 6]`.

(b)
```
def first_minus_second_no_obj(L1, L2):
    """Removes elements of one list in accordance with another.

    No new objects (except ints) are created during the function call!!
    None is returned, and L2 is unaffected by the function call,
    but L1 is mutated by the function call.

    At the end of the function call,
    all elements of L1 that occur in L2 have been removed.
    The remaining elements are left in their original order.
    """

    pass
```

Test:

```
L1 = [8,8,1,2,7,3,4,7,5,6,11,8,13,8,12,8,11,8,18,18]
L2 = [7,11,19,19,19,13,13,12,12]
first_minus_second_no_obj(L1, L2)
print(L1)
```

should print `[8, 8, 1, 2, 3, 4, 5, 6, 8, 8, 8, 8, 18, 18]`.

(Similar comments about "no new objects" to before.)

(c)
```
def keep_last_occurrences_no_obj(L):
    """Deletes duplicates in a list of numbers.

    No new objects (except ints) are created during the function call!!
    None is returned, but L is mutated by the function call.

    At the end of the function call,
    L consists of the same numbers as it began with in the same order.
    However, all but the last occurrence of each number has been deleted.
    """

    pass
```

Test:

```
L = [1,2,3,1,4,1,5,2,6,6,6,5,3,2,2,2,2,3,1,1,1,1,1,4,3,1,2,1]
keep_last_occurrences_no_obj(L)
print(L)
```

should print `[6, 5, 4, 3, 2, 1]`.

(d)
```
def keep_first_occurrences_no_obj(L):
    """Deletes duplicates in a list of numbers.

    No new objects (except ints) are created during the function call!!
    None is returned, but L is mutated by the function call.

    At the end of the function call,
    L consists of the same numbers as it began with in the same order.
    However, all but the first occurrence of each number has been deleted.
    """

    pass
```

Test:

```
L = [1,2,3,1,4,1,5,2,6,6,6,5,3,2,2,2,2,3,1,1,1,1,1,4,3,1,2,1]
keep_first_occurrences_no_obj(L)
print(L)
```

should print `[1, 2, 3, 4, 5, 6]`.

3.
```
def primes(N):
    """Returns a list of the first N prime numbers.

    Receives an int N bigger than or equal to 0,
    and does what it says above.
    """

    pass
```

Test: `print(primes(8))` should print `[2, 3, 5, 7, 11, 13, 17, 19]`.