Question 1:

Consider the following two implementations of a function that if given a list, lst, create and return a new list containing the elements of lst in reverse order.

If lst is a list of n integers,

- 1. What is the worst case running time of reverse1(lst)? Explain of your answer.
- 2. What is the worst case running time of reverse2(lst)? Explain of your answer.
- 1) The worst case running time of reversel (1st) is $\theta(n^2)$. The insertion of elements at the beginning of the list is ultimately a $\theta(n)$ operation and is performed $\theta(i)$ times.
- 2) The worst case running time of reverse 2(1st) is $\theta(n)$. Instead of inserting at the beginning of the list, the elements are appended in reverse order. Appending has a runtime of $\theta(1)$ and this operation is performed $\theta(i)$ times

```
3(b) def find_duplicates (1st):
     O(1) [dups = []
len_ls+ = len(ls+)
            for i in range (len-1st): (0(i)
                if 1s+ [abs (1s+[i])] >= 0:
                     1s+ [abs (1s+[i])] = -|s+[abs(1s+[i])]
                else:
                   dups.append (abs (1s+[i])) (1)
    O(1) / return dups
             Worst case running time: O(n)
 4(0)
           def remove_all(1st, value):
         \theta(1) \begin{cases} 1-1s+ = len(1s+) \\ c+r = 1 \end{cases}
                 for x in range (1-1st): (x)
                 OLI [if Is+[x] != valve:
                                                          0(zx)
                    O(1) [ |s+[c+r] = |s+[x]
c+r+=1
                for x in range (ctr, 1-1st): WO(x)
               O(1) [ 1st.popc)
                worst case rung time: O(n)
```

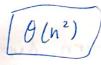
Question 4:

The remove (value) method of the list class, removes the **first** occurrence of value from the list it was called on, or raises a ValueError exception, if value is not present.

Note: Since remove needs to shift elements, its worst-case running time is linear.

In this question we will look into the function remove_all(lst, value), that removes all occurrences of value from lst.

a) Consider the following implementation of remove_all:



Analyze the worst-case running time of the implementation above.

- b) Give an implementation to remove_all that runs in worst-case linear time.

 Notes:
 - 1. Your implementation should **mutate the given list object** (in-place), without using an additional data structure.
 - 2. Your implementation **should keep the relative order** of the elements that remain in the list
- c) Analyze the worst-case running time of your implementation in (b).