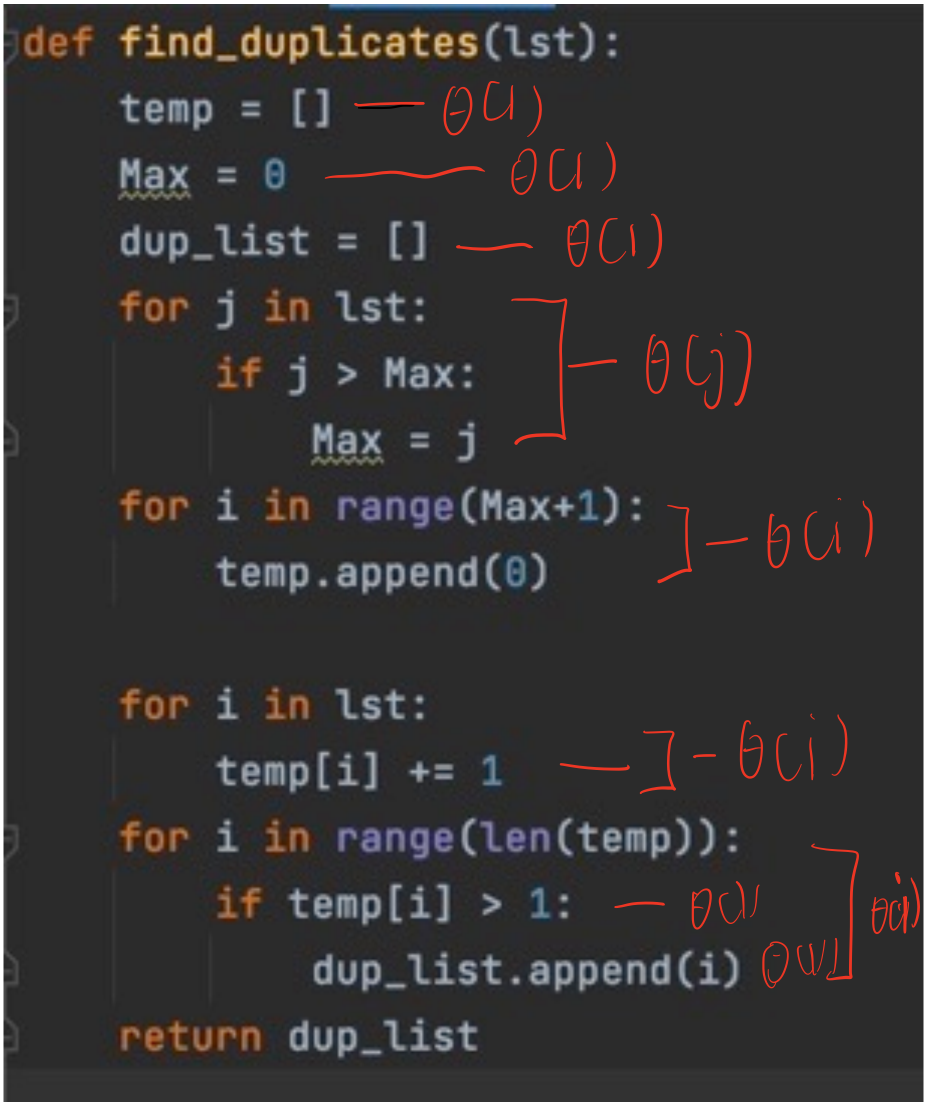
1. the worst case running time of reversed(lst) is $\theta(n^2)$. Everytime we insert a element, we need to plus 1 on the index of the element we insert. Thus the running time of insert 1 element is $\theta(n)$. As we insert n elements, the running time would become $\theta(n^2)$

2. the worst case running time of reverse2(lst) is $\Theta(n)$. The running time of append one element to the lst is $\Theta(1)$. As we append n elements, the running time would become $n * 1$ which equal to $\Theta(n)$

3. b.



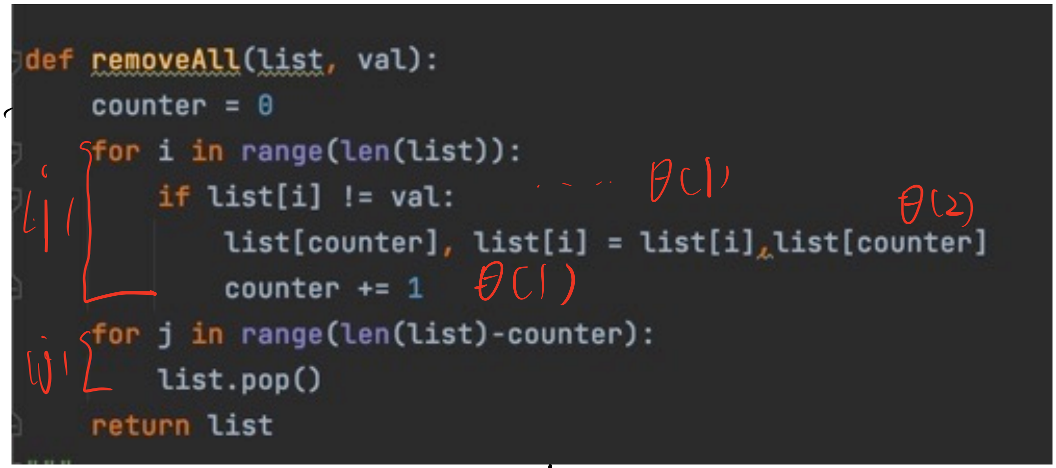
```
def find_duplicates(lst):
    temp = []
    Max = 0
    dup_list = []
    for j in lst:
        if j > Max:
            Max = j
    for i in range(Max+1):
        temp.append(0)

    for i in lst:
        temp[i] += 1
    for i in range(len(temp)):
        if temp[i] > 1:
            dup_list.append(i)
    return dup_list
```

The worst-case would "lst" contain n identical numbers. At that time, the running time would become $n + n + 2n + 4 = \Theta(n)$

4.a. The worst case would be "lst" contain n numbers of the value which needs to be removed. ~~At~~ that time, the runing time would become $n * n$ $= \Theta(n^2)$

4 C



```
def removeAll(list, val):
    counter = 0
    for i in range(len(list)):
        if list[i] != val:
            list[counter], list[i] = list[i],list[counter]
            counter += 1
    for j in range(len(list)-counter):
        list.pop()
    return list
```

The worst case would be the "list" Conta n numbers of the value which needs to be removed At that time, the running time would become

$n+n = \Theta(n)$