tg1632_hw4.pdf

3c) 1. n appends and n pops
append is $O(1)$ hence n appends is $O(n)$
pop is $O(1)$ hence n pops is $O(n)$
$O(n) + O(n) = O(2n) \Rightarrow O(n)$

2. $\Omega(n^2)$ means n^2 is the lower bound
when the array goes below $\frac{n}{2}$.
Resize = $O(n)$
$\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \ldots = X$
Geometric sequence, $a = \frac{n}{2}$ $r = \frac{1}{2}$ $X = \frac{n/2}{1-\frac{1}{2}} = n$
$O(n)$, n times $\rightarrow O(n \cdot n) = \boxed{O(n^2)}$

4b)
```
def find_duplicates(lst):
    dup = []
    for i in range(len(lst) - 1):
        x = abs(lst[i])          → O(1)
        if (lst[x] > 0):
            lst[x] = -lst[x]     → O(1)
        else:
            dup.append(-1 * lst[x])
    return dup                   → O(1)
```
loops n times

This has worse-case running time $\boxed{O(n)}$.

5a)
```
def remove_all(lst, value):
    cnd = False
    while (cnd == False):
        try:
            lst.remove(value)
        except ValueError:
            cnd = true
```

✡Worst case where lst is a lst where every element is value. Has to remove n times. Remove is $O(n)$. Therefore it is $O(n \times n) = \boxed{O(n^2)}$

5c) def remove_all(lst, value):

```
    x = len(lst)
    for i in range(len(lst)):        → loops n times
        if lst[i] != value:
            lst.append(lst[i])       → O(1) time
    lst[:] = lst[x:]                 → O(n) time
```

$O(n + 1 + n) \Rightarrow \boxed{O(n)}$