Problem Set 5

Assignement 1

1. In the worst case will be n, cause it needs to go through every item. If the list were sorted only 1.
2. Worst case will be O(n/2).
3.
    a. In the worst case we go through every element resulting in **n complexity.**
    b. If the element is in the first position, will be o(1)
4.

__init__	O(1)
__len__	O(1)
__contains__	O(n)
add	O(n)
valueOf	O(n)
remove	O(n)
__iter__	O(n)
_findPosition	O(n)

5. Worst case is $o(n^2)$
   Best case is o(n)

Assignment 2

1. See python file.
   The complexity remains o(log n) because we only need to add a statement to check whether it's the first occurrence or not.

2. The runtime is o(n) as it has to go through every element.
3. See python file.

Assignment 3

1. Sorting (80, 7, 24, 16, 43, 91, 35, 2, 19, 72)
    a. Bubble sort
       (80, 7, 24, 16, 43, 91, 35, 2, 19, 72)
       (7, 24, 16, 43, 80, 35, 2, 19, 72, 91)
       (7, 16, 24, 43, 35, 2, 19, 72, 80, 91)
       (7, 16, 24, 35, 2, 19, 43, 72, 80, 91)
       (7, 16, 24, 2, 19, 35, 43, 72, 80, 91)
       (7, 16, 2, 19, 24, 35, 43, 72, 80, 91)
       (7, 2, 16, 19, 24, 35, 43, 72, 80, 91)
       (2, 7, 16, 19, 24, 35, 43, 72, 80, 91)
       (2, 7, 16, 19, 24, 35, 43, 72, 80, 91)
       (2, 7, 16, 19, 24, 35, 43, 72, 80, 91)

b. Selection sort
(80, 7, 24, 16, 43, 91, 35, 2, 19, 72)
(2, 7, 24, 16, 43, 91, 35, 80, 19, 72)
(2, 7, 24, 16, 43, 91, 35, 80, 19, 72)
(2, 7, 16, 24, 43, 91, 35, 80, 19, 72)
(2, 7, 16, 19, 43, 91, 35, 80, 24, 72)
(2, 7, 16, 19, 24, 91, 35, 80, 43, 72)
(2, 7, 16, 19, 24, 35, 91, 80, 43, 72)
(2, 7, 16, 19, 24, 35, 43, 80, 91, 72)
(2, 7, 16, 19, 24, 35, 43, 72, 91, 80)
(2, 7, 16, 19, 24, 35, 43, 72, 80, 91)

c. Insertion sort
(80, 7, 24, 16, 43, 91, 35, 2, 19, 72)
(7, 80, 24, 16, 43, 91, 35, 2, 19, 72)
(7, 24, 80, 16, 43, 91, 35, 2, 19, 72)
(7, 16, 24, 80, 43, 91, 35, 2, 19, 72)
(7, 16, 24, 43, 80, 91, 35, 2, 19, 72)
(7, 16, 24, 43, 80, 91, 35, 2, 19, 72)
(7, 16, 24, 35, 43, 80, 91, 2, 19, 72)
(2, 7, 16, 24, 35, 43, 80, 91, 19, 72)
(2, 7, 16, 19, 24, 35, 43, 80, 91, 72)
(2, 7, 16, 19, 24, 35, 43, 72, 80, 91)

2. To not repeat the same output every time, it will be the same for n-1 iterations because it's already sorted.

Assignment 4

See python file.