Problem Set 5, Part I

Problem 1: Sorting practice

1-1) `{3, 4, 18, 24, 33, 40, 8, 10, 12}`

1-2) `{4, 10, 18, 24, 33, 40, 8, 3, 12}`

1-3) `{4, 10, 18, 8, 3, 12, 24, 33, 40}`

1-4) `{10, 18, 4, 24, 12, 3, 8, 40, 33}`

1-5) `{10, 18, 4, 3, 12, 24, 8, 40, 33}`

1-6) `{4, 10, 18, 24, 33, 40, 8, 3, 12}`

Problem 2: Practice with big-O

2-1)

function	**big-O expression**
a(n) = 5n + 1	a(n) = O(n)
b(n) = 5 - 10n - n^2	b(n) = O(n^2)
c(n) = 4n + 2log(n)	c(n) = O(n)
d(n) = 6nlog(n) + n^2	d(n) = O(n^2)
e(n) = 2n^2 + 3n^3 - 7n	e(n) = O(n^3)

2-2) O(n^2). The runtime of the outermost loop is 3 and the middle loop is n and the inner loop is n since it counts 0 to n until n times. So run-time is 3 * n * n = 3n^2. So the run time is n^2

2-3)O(nlogn)For 0 to n is n times and n to 0, divided by integer division 2 is log n with base 2. So the O(n) run time is n * log2n = nlog(n).

Problem 3: Comparing two algorithms

worst-case time efficiency of algorithm A: ***(n-1) * (n/2) = n^2/2 - n/2 = O(n^2)***

Explanation: ***Basically run the whole algorism from start with n-1 when i = 0 until 0 when i adds up to array's length - 1. So there are total n number of i and j is decreasing by 1 for each j start from n-1 when i = 0. So Addition of n-1, n-2, n-3, … , 2, 1, 0 is the multiple of (n - 1) and (n/2).***

worst-case time efficiency of algorithm B: ***O(n*logn)***

Explanation: ***This algorithm contains Mergesort and its' own for loop. The worst case of mergesort and the best case of runtime is O(nlogn) no matter what. And it takes n times***

to search duplicate through from the beginning until the end. So it would takes O(nlogn + n) = O(nlogn)

Problem 4: Practice with references

4-1)

Expression	**Address**	**Value**
n	0x100	0x712
n.ch	0x712	'n'
n.prev	0x712 + 6 = 0x718	0x064
n.prev.prev	0x064 + 6 = 0x070	0x360
n.prev.next.next	0x712 + 2 = 0x714	null
n.prev.prev.next	0x360 + 2 = 0x362	0x064

4-2)

```
n.prev.next = x;
x.next = n;
x.prev = n.prev;
n.prev = x;
```

4-3)

```
public static void initPrevs(DNode first){
      DNode trav = first;
      DNode trail = null;

      while(trav != null){
            trail = trav.next;
            trail.prev = trav;
            trav = trav.next;
```