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, 8, 12, 3, 24, 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)
This is because there are three nested for loops which depend on the previous for loop for their length. However, the first for loop only repeats three times regardless of n and hence does not really matter in the context of Big O.

2-3)

O(n^2)
Technically, the nested for loops should give you a O(n * n/2) due to the second loop only iterating for half of the n elements. However, this simplifies to O(n^2) since the expression above is closer to O(n^2) than to O(n). You can see this more clearly in a graphing software where n*n/2 is a lot closer to n^2 than to n.

Problem 3: Comparing two algorithms
worst-case time efficiency of algorithm A: O(n^2)
*Explanation: Since the outer for loop can traverse from 0 to n-1,the inner loop can go from i+1 to n, and the inner loop is nested within the outer loop, then we can get the expression of O(n*n) which can then just be simplified to O(n^2).*

worst-case time efficiency of algorithm B: O(nlogn)
Explanation: The first mergesort call has a worst case scenario of running at O(nlogn). The loop after that runs from 1 to n which yields a running time of O(n). Combining these we get the expression O(n + nlogn) which can then be simplified to O(nlogn).

Problem 4: Practice with references
4-1)

Expression	Address	Value
n	0x100	0x712
n.ch	0x712	‘n’
n.prev	0x718	0x064
n.prev.prev	0x070	0x360
n.prev.next.next	0x714	null
n.prev.prev.next	0x362	0x064

4-2)

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

4-3)

```
public static void initPrevs(DNode one){
DNode trav = one;
DNode trail = null;
while (trav != null && trav.ch <ch) {
     trav.prev = trail;
     trail = trav;
     trail = trail.next;
}
}
```