Clicker Qs
Give a Big Oh analysis of this loop:

```java
int sum = 0;
for (int i = 0; i < N; i++) {
    sum = sum + i*N;
    for (int j = 0; j < 10; j++)
        sum = sum - j;
}
System.out.println(sum);
```

A. O(1)  
B. O(N)  
C. O(10*N)  
D. O(N^2)
Answer B: $O(N)$ The inner loop (on $j$) is a constant number of steps.
Give a Big Oh analysis of this loop:

```java
int sum = 0;
for (int i = 0; i < N; i++) {
    sum = sum + i*N;
    for (int j = 0; j < N; j++)
        sum = sum – j;
}
System.out.println(sum);
```

A. O(1)
B. O(N)
C. O(10*N)
D. O(N^2)
Answer D: $O(N^2)$ The inner loop (on j) takes N steps and we do it once for each value of i. The whole thing takes $N+N+N+...+N$ steps. There are N values of i, so this is $N*N$ or $N^2$. 
Give a Big Oh analysis of this loop:

```java
int sum = 0;
for (int i = 0; i < N; i++) {
    sum = sum + i*N;
    for (int j = 0; j < i; j++)
        sum = sum - j;
}
System.out.println(sum);
```

A. $O(1)$
B. $O(N)$
C. $O(10*N)$
D. $O(N^2)$
Again the answer is D: $O(N^2)$. The loop on j runs for i steps so this takes $0 + 1 + 2 + 3 + \ldots + (N-1)$. You might know a formula for a sum like that; its sums to $(N-1)*N/2$. If you don’t know the formula think about it this way. We are adding N numbers all of which are smaller than N, so the sum is less than $N*N$. On the other hand half of the numbers ($N/2$ of them) are all larger than $N/2$, so the sum is more than $(N/2)*(N/2)$. So the sum is more than $N^2/4$ and less than $N^2$; it must be $O(N^2)$. 
Here’s a fast way to sort a list of N integers if they come in a fixed range. For simplicity suppose they are all between 0 and 9. Note that the sorted list will look like 0 0...0 1 1...1 2.....9

Let’s call the original list L.

Make an array int A = new int[10] with all of its entries initialized to 0. For each i A[i] will count how many times the number i appears in the list.

Loop through all the entries of the list. If an entry is i then do

A[i] += 1

k = 0;
for (int i = 0; i < 10; i++) {
    for (int j = 0; j < A[i]; j++) {  // this puts A[i] i’s into L
        L.set(k, i);
        k += 1;
    }
}
So we have two loops. The first loop walks through the list counting how many 0’s, how many 1’s etc it contains.
The second loop says: Oh, we need this many 0s, so it sets the first that many entries of the list to 0. Then its says We need that many 1s, so it sets the next that many entries to 1. And so forth for all 10 digits.

If the original list has N entries, how long does this take?

A. $O(10) = O(1)$
B. $O(N)$
C. $O(N^2)$
D. $O(N^3)$
This is $O(N)$; both loops do $N$ steps and they are sequential. We do the $N$ steps of the first loop than the $N$ steps of the second loop, for a total of $2*N$ steps.