While Loop Analysis

1. Analyze the following algorithm. Note that I assign n to a local variable, i, here to avoid confusion on problem size n.

```
int example1(int n)
{
  i = n;
  while (i>0) {
    work in constant;
    i = i/3;
  }
}
```

   Take the "work in constant" statement as the barometer instruction. The analysis is indeed to count how many times this statement is executed, or in other words, how many times the loop body is executed. Note that i gets initial value n and is divided by 3 iteratively. For simplicity, we can assume $n = 3^k$ for some k. Now i is initialized to 3^k. At the end of the first loop iteration, i is reduced to 3^{k-1} which is the value of i at the beginning of the second iteration. At the end of the second iteration, i is reduced to 3^{k-2}. Following this trend, we can see that i is reduced from $3^1 = 3$ to $3^0 = 1$ in the k_{th} iteration. Loop iterates one more time when $i = 1$ becomes $i = \lfloor 1/3 \rfloor = 0$. Then the next loop test fails. So the total number of loop iterations is $k + 1$. Note that $k = log_3 n$, which gives the total number of iterations as $log_3 n + 1 \in \Theta(log n)$.

   If n is not an exact power of 3, the analysis is similar. The loop iterates until i becomes $\frac{n}{3^{\lfloor log_3 n \rfloor}} = 1$. And one more iteration makes i 0. So the total loop iterations is $\lfloor log_3 n \rfloor + 1 \in \Theta(log n)$.

   For this course, you are allowed to assume that n is an exact power of 3. However, as you can see, it does not change the asymptotic bound for this category of loops no matter n is an exact power or not.

   An alternative method for analyzing this loop is to come out a recurrence equation for the number of loop iterations. Let $L(n)$ be the total number loop iterations. Note that the number of loop iterations after the first round is equivalent to the case when i is initialized to $\lfloor n/3 \rfloor$. So we get $L(n) = L(\lfloor n/3 \rfloor) + 1$. We know $L(1) = 1$. Solving this recurrence, we get $L(n) \in \Theta(log n)$ based on the Master theorem which will be covered next week.

2. Analyze the following algorithm. Note that I assign n to a local variable, j, here to avoid confusion on problem size n.

```
int example2(int n)
{
  j = n;
  while (j>0) {
    for (i=0; i<j; i++) {
      work in constant;
    }
    j = j/3;
  }
}
```

   Take the "work in constant" statement as the barometer instruction. The analysis is indeed to count how many times this statement is executed, or in other words, how many times the inner loop body is executed. Note that for each outer loop iteration, j, the inner loop iterates j times. Again we can assume that $n = 3^k$. Based on the analysis of the first example, we note that the value of j starts as 3^k, is divided by 3 at each outer

loop iteration, and reaches $3^0 = 1$ at the beginning of the last iteration. So the total inner loop iterations is $\sum_{l=0}^{k} 3^l = \frac{3^{k+1}-1}{3-1} = \frac{3n-1}{2} \in \Theta(n)$.

We can also build a recurrent equation for this problem. Let $L(n)$ be the total number of inner loop iterations. We have $L(n) = L(\lfloor n/3 \rfloor) + n$ and $L(1) = 1$. The solution to this recurrence is $L(n) \in \Theta(n)$ based on the Master theorem.