Problem #1: Indicate the growth rates (big-O) for the following code fragments. Detailed calculations are not necessary. Based on the info provided in class you should be able to “glance” at these and provide an answer.

(a)  
```
sum = 0;
for(i=0; i<n; i++)
    for(j=n; j>0; j--)
        sum++;
```

(b)  
```
sum = 0;
for(i=0; i<n; i++)
    for(j=n; j>0; j--)
        sum += 328 * n % (i+1);
```

(c)  
```
sum = 0;
for(i=0; i<n; i++)
    for(j=0; j<n*n; j++)
        for(k=0; k<n*n; k++)
            sum++;
```

(d)  
```
sum = 0;
i = 0;
while(i<n)
{
    for(j=0; j<n; j++)
    {
        sum++;
    }
    i++;
}
```

(e)  
```
sum = 0;
i = 0;
while(i<n)
{
```
for(j=0; j<n; j++)
{
    if(j%2 == 0)
    {
        for(m=0; m<n; m++)
        {
            sum++;}
    }
}
i++; (f)
sum = 0;
for(i=0; i<n; i++)
    for(j=1; j<10; j++)
        sum++;;

Problem #2: One at a time, put each code fragment in problem #1 into a complete program. Give the running time for at least 5 values of n (using a stopwatch or time related functions specific to the programming language). Compare your Big-O analyses in problem #1 with the actual running times by plotting your observed values. Then on top of that data, also plot the curve for the expected big-O. They should line up pretty nicely. If your data does not show a clear trend, then plot more data that is spread out over a wider range of values.

Hint: Use values of n that give reasonable times. In most cases, that will mean using very big values of n like 10,000 or larger. Smaller values will usually run faster than can be accurately measured.

Another hint: Remember, in Java, “System.currentTimeMillis()” will give the current time in milliseconds (measured from midnight, Jan. 1, 1970).

Another hint: Suppose that you expect something to run in O(N²). Then you will want to plot cN² for some value of c. Plot this on top of your data. They should line up nicely. But what value of c should you use? Well, play around and choose a good fit! Or if you know how, have Excel (or some other program) do a regression to find a reasonable value for c.

Another Hint: You may want to come into the CS lab and run your programs on lots of computers simultaneously so that you can finish faster!

Extra Credit (5 points): Consider the recursive algorithm for calculating x^n. Calculate how many multiplications happen for each value of n = 1 to n = 40.
Using a plotting program (such as Excel), plot \( n \) versus the number of multiplications. For some constants \( a \) and \( b \), does \( "a \log(n) + b" \) match the data? Is this what you expected? Why?

If you understand why, then you are in good shape for the midterm. **This would make a good big-O study problem for the midterm!**