Name(s): ____

© Rosanna Heise

UNIVERSITY OF ALBERTA

AUCSC 112 LAB

Assignment #4

(Due before midnight on day before the next lab)

Goals: We want our computer programs to run fast, so we must be able to:

1. Understand where complexity lies in algorithms.
2. Calculate complexity exactly and in general terms using order notation.
3. Read Java code and determine what it does

.

Reference:

Heise, Chapter 4.

Instructions:

For this week's assignment you may work as an individual, or a group of two. Do not share any answers between groups. If working in a group, each person must solve and understand together with the other person – do not split the work up and assign parts to each person. Your partner must come from your lab section.

Complete the following questions, using Word or neat hand-printing. ***Show all work.*** Explain your answer to each question completely – there are few marks for a number or an equation without explanation.

Questions:

1. Given the following Java code:

```
int answer = 0;
while (n > 1){
   answer = answer + n;
   n = n / 2;
}
```

a. Complete the following table to indicate how many assignment operations occur in this piece of code, for the given values of *n*:

n	Number of assignments
15	
16	
60	
85	
100	
1000	

b. Find a general formula for the number of assignment operations. Explain each part of the formula using your own words.

c. Use your formula to find the number of assignment operations when *n* is one million.

d. If each assignment operation takes half a nanosecond to complete, how long will the loop take to run when *n* is one million?

e. Characterize the run-time complexity of this code using order notation.

2. Given the following Java method:

```
public void doSomething(int n){
   for (int i = 0; i < n; i += 2){
      for (int j = 1; j < n * n * n; j *= n){
         for (int k = 1; k < n; k *= 3){
            expensiveOp();
         }
      }
   }
}
```

a. Complete the following table, assuming a call to `doSomething(n)` occurred and determining how many times each loop body runs, and ultimately how many calls are made to `expensiveOp`.

n	Inner-most loop (`k`)	Middle loop (`j`)	Outer-most loop (`i`)	`expensiveOp`
3				
5				
10				
60				
85				
100				

b. Determine an exact formula for the number of calls to `expensiveOp` for a general value of n > 1. Explain, in your own words, what each part of the formula represents. (You may need to use the floor or ceiling functions of math.)

c. Use your formula to determine the number of calls to `expensiveOp` when $n = 10923$.

d. Express your formula from part b in order notation.

3. There are two software applications, A and B, which do the same thing and fall within the same price point. It is known that application A has a run-time of $0.12n \log_2 n$ milliseconds while application B has a run-time of $2.56\, n$ milliseconds.

a. Which software application is better? Explain fully.

b. Characterize the run-time of each application using order notation.

4. Given the following Java code:

```
int x = 0;
for (int bog = 2; bog < n; bog += 2){
   for (int mar = 1; mar < bog; mar++){
      x = x + mar * bog;
   }
}
```

a. Complete the following table to indicate how many multiplications occur for the given value of *n*:

n	Number of multiplications
2	
5	
15	
20	
25	

b. Find a general formula for the number of multiplications. Explain each part of your formula.

c. Characterize the complexity of the code using order notation. Explain why this characterization is meaningful, and compare to what you found in part b.