Problem 1. (10 points) Go to the Piazza discussion board (https://piazza.com/ucr/fall2014/cs141/home or follow the link from the class CS 141 webpage), register yourself, then select hw1 from the left upper-corner and post one message to introduce yourself.

Problem 2. (20 points)
Analyze the worst-case time complexity of the following method, and give a tight bound using the big-theta notation

Algorithm WeirdLoop (n : integer)
for i ← 1 to n log n do
    j ← i
    while j ≤ n do
        “loop body”
        j ← j + 1
assuming that “loop body” takes Θ(1) time.

Answer: When i ≤ n, both “loop body” and the j ← j + 1 statement are executed n + (n − 1) + . . . + 2 + 1 = n(n + 1)/2 ∈ Θ(n²) times. Since “loop body” takes constant time, so the cost of the loop when i ≤ n is Θ(n²) time. The outer for-loop takes Θ(n log n) time. Overall, the algorithm takes Θ(n²) time.

Problem 3. (20 points) Order the following list of functions by the big-Oh notation, i.e., rank them by order of growth. Group together (for example, by underlining) those functions that are big-Theta of one another. Logarithms are base two unless indicated otherwise.

\[
\begin{array}{cccccccc}
3n + 2 & n^3 + n^2 & \log^2 n & \log \log n & 2^n & 4^{n-1} \\
2\log n & 2\sqrt{2\log n} & \sqrt{n} & n^3 & 1 & 3^{n/3} \\
10000n^2 & 2^{n+1} & e^n & n\log\log n & \log n & (\log n)^2 \\
2^n & n3^n & 4^{\log n} & 4n^2/\sqrt{n} & \sqrt{\log n} & n\log n \\
\end{array}
\]

Answer: Here are the function in decreasing order of asymptotic growth (fastest growing to slowest growing)

\[
\begin{array}{cccccccc}
2^{2n+1} & 2^{2n} & 4^{n-1} & n3^n & e^n \\
2^n & 3^{n/3} & n\log\log n & \{n^3 + n^2, n^3\} & \{10000n^2, 4^{\log n}\} \\
4n^2/\sqrt{n} & n\log n & \{3n + 2, 2^{\log n}\} & \sqrt{n} & 2\sqrt{2\log n} \\
\{\log^2 n, (\log n)^2\} & \log n & \sqrt{\log n} & \log \log n & 1 \\
\end{array}
\]

Functions in \{\} are Θ of each other.
Problem 4. (50 points)
You are facing a high wall that stretches infinitely in both directions. There is a door in the
cell, but you don’t know how far away or in which direction. It is pitch dark, but you have a very
dim lighted candle that will enable you to see the door when you are right next to it. Show that
there is an algorithm that enables you to find the door by walking at most $O(n)$ steps, where $n$ is
the number of steps that you would have taken if you knew where the door is and walked directly
to it. What is the constant multiple in the big-O bound for your algorithm?

Answer: First, note that even if we knew the true distance $n$, we would need $3n$ steps in the worst
case. Can linear-time be achieved and how bad is the constant when we do not know $n$?

Consider the following algorithm.
1. $k \to 1$
2. take $k$ steps on left
3. if door found then STOP
4. take $k$ steps on the right (back to the origin)
5. $k \to 2 \cdot k$
6. take $k$ steps on the right
7. if door found then STOP
8. take $k$ steps on the left (back to the origin)
9. $k \to 2 \cdot k$
10. goto 2.

Analysis: First, assume that $n = 2^q$. Note that the worst case is when you travel the first time $2^q$
steps left and the door is $2^q$ steps on the right of the origin. In that case you will travel $2^q$ to the
left, come back to the origin, walk another $2^q$ to the right, and finally reach the door. Then, the
number of steps would be
\[
(1 + 1 + 2 + 2 + \ldots + 2^q + 2^q) + 2^q = 2 \left(2^{q+1} - 1\right) + 2^q = 5 \cdot 2^q - 2 = 5n - 2
\]

In general, however, $n$ is not a power of two. Let us set $q = \lfloor \log_2 n \rfloor$. Note that $2^q < n < 2^{q+1}$.
Because we did not find the door at $2^q$, the algorithm will go back to the origin, go in the other
direction $2^{q+1}$ steps, come back, and finally travel $n$ positions to the door. Therefore, we have
\[
1 + 1 + 2 + 2 + \ldots + 2^q + 2^q + 2^{q+1} + 2^{q+1} + n = 2 \left(2^{q+2} - 1\right) + n = 8 \cdot 2^q - 2 + n = 9n - 2
\]

So the constant is 5 when $n$ is a power of two, 9 otherwise (worst-case).

The complexity of the algorithm is $O(n)$, although different algorithms may result in different
constants.