CS251 Week 04 Lab Exercises: SAMPLE SOLUTIONS
More Runtime Practice!

PART 1: spend about 20 minutes working through these exam-style questions to the best of your ability. Debate your answers with your neighbors.

QUESTION 1.1:

Fill in the blanks. Complete each of the statements below with either O, Ω, or Θ whichever is correct and gives the most information about the relationship.

$n^2/10 - 100n$ is __Ω_____($n \log n$)

$log\ n$ is ____O_______($(\log n)^2$)

$n^2 \log n$ is ___Ω______(n^2)

$n!$ is ____Ω______($2^n/20$)

$\log n$ is _____Θ_______($\log(n^2)$)

2^n is _____O________(3^n)

$\log(2^n)$ is _______Θ_______($n/5$)

QUESTION 1.2:

We are given the following facts:

“Algorithm A’s worst case runtime is $O(n^2)$”

“Algorithm B’s worst-case runtime is $O(n^3)$”

As far as worst-case asymptotic runtime, which algorithm would we prefer?

a. Algorithm A
b. Algorithm B
c. not enough information - since big-Oh is an upper-bound, we don’t know if these bounds are loose or tight from the given info.
d. depends on the value of n

QUESTION 1.3: suppose I have two algorithms X and Y and I know that the **best, worst and average case runtime** properties below hold:

“Algorithm X’s runtime is
$\Theta(n^2)$”

“Algorithm Y’s runtime is $\Theta(n \log n)$”

TRUE or FALSE: “Algorithm Y performs faster than Algorithm X **for all input sizes”**

a. TRUE
b. FALSE - for small inputs, algorithm X could be faster because of a smaller runtime coefficient. This is typically the case with insertion-sort versus the asymptotically faster merge-sort

PART 2: After some TA-led discussion, now work through as many of the following as you can.

QUESTION 2.1:

Which statement best describes the **worst-case** runtime of the C function below as a function of n?

```
int foo(int a[], int n) {
int i, j, x;

  x=0;
  for(i=0; i<n; i++) {
      if((a[i] % 2) == 0){
            for(j=1; j<n; j *=2){
                 x += a[j];
            }
      }
      else {
            for(j=0; j<i; j++) {
                 x -= a[j];
            }
      }
     return x;
}
```

a. $\Theta(n \log n)$
b. $\Theta(n^2)$
c. $O(n \log n + n^2)$
d. $O(n)$
e. $\Theta(n^2/2)$

QUESTION 2.2: Same function; which statement best describes the **best-case** runtime of foo?

```
int foo(int a[], int n) {
int i, j, x;

  x=0;
  for(i=0; i<n; i++) {
      if(a[i] % 2) == 0){
            for(j=1; j<n; j *=2){
                 x += a[j];
            }
      }
      else {
            for(j=0; j<i; j++) {
                 x -= a[j];
            }
      }
     return x;
}
```

a. $O(n^2)$
b. $\Omega(n)$
[✓ c. $\Theta(n \log n)$]
d. $\Theta(n^2)$

QUESTION 2.3: consider the C function `tick` below. If we call **`tick(4),`** how many times will **`TICK`** be printed to the screen?

```
void tick(int n) {
int i;

    if(n <= 0) return;

    for(i=0; i<n; i++){
        printf(“TICK\n”);
    }
    tick(n/2);
}
```

a. 4
b. 16
c. 8
[✓ d. 7]

QUESTION 2.4: same function. How many TICKs are printed when `tick(8)` is called?

a. 8
b. 24
[✓ c. 15]
d. 16

QUESTION 2.5: same function. Let $T(n)$ be the number of TICKs printed when tick called with a parameter of n. Which of the following best describes $T(n)$

a. $T(n)$ is $\Theta(n\log n)$
b. $T(n)$ is $\Theta(n^2)$
c. $T(n)$ is $\Theta(2n)$
[✓ d. $T(n)$ is $\Theta(n)$]