COMP 53: Algorithm Analysis, Part 1

Instructions: In this lab, we are going to review Constant time operations, growth of functions, complexity and big O notation.

- Get into groups of **at most two people** to accomplish this lab.
- At the top of your submitted file list the group members as a comment.
- Each member of the group must individually submit the lab in Canvas.
- This lab includes **40 points** in aggregate. The details are given in the following.

1 Constant time operations

Specify whether the following operations are constant time or non-constant time.

1. `int x = 5; int y = x;`
2. ```
   if (x < n) {
           x = n % 2;
   }
   ```
3. `strcmp(s1, s2);`
4. ```
   // n is the input to the program
   for (int i = 0; i < n; i ++)
           x++;
   ```
5. ```
   for (int i = 0; i < 25; i ++)
           x++;
   ```
6. ```
   // n is the input to the program
   x = n * x - 12;
   ```
7. ```
   // n is the input to the program
   a [23] = n * 5;
   ```

2 Growth Rate

Assume that an algorithm's runtime growth rate is $12n^2 + 15$ in the best case, and $3n^2\ log\ n$ in the worst case. Which ones are lower bound for this algorithm? Which ones are upper bound for this algorithm?

1. $35n$
2. $13n$
3. $10n^2$
4. n^3
5. $3n^3$
6. $15n\ log\ n$
7. $12n^2 + 15$
8. $3n^2\ log\ n$
   ```

3 Asymptotic Notations

Consider the runtime growth rate $T(n) = 12n^2 + 3n + 15$. Specify whether the following hold or not.

1. $T(n) = O(n)$
2. $T(n) = O(n \; log \; n)$
3. $T(n) = O(n^2)$
4. $T(n) = O(n^3)$
5. $T(n) = \Omega(n)$
6. $T(n) = \Omega(n \; log \; n)$
7. $T(n) = \Omega(n^2)$
8. $T(n) = \Omega(n^3)$
9. $T(n) = \Theta(n)$
10. $T(n) = \Theta(n \; log \; n)$
11. $T(n) = \Theta(n^2)$
12. $T(n) = \Theta(n^3)$

Determine the most simplified O notation for each of the following.

1. $O(3n + 5)$
2. $O(3n + 4n^2)$
3. $O(1 + 0.5n)$
4. $O(1) + O(3n + 5)$
5. $O(n + 5) \times O(4n + 3n \; log \; n)$
6. $O(1 + 5n) + O(2 + 3n)$

4 Runtime Analysis of Nested Loops

Determine the worst time runtime for each of the following.

1. ```
   for(i = 0; i < n; i++)
       x = x - 3;
   ```

2. ```
   for(i = 0; i < n; i++) {
       x = x - 3;
       y--;
   }
   ```

3. ```
   for(i = 1; i < n; i *= 2) {
       x = x - 3;
       y--;
   }
   ```
   ```

4.
```
for(i = 1; i < n; i *= 2) {
        for (j = 0; j < n; j++) {
                x++;
                if (x < y)
                        x = 4;
        }
}
```

5.
```
for(i = 1; i < n; i *= 2) {
        for (j = i; j < n; j++) {
                x++;
                if (x < y)
                        x = 4;
        }
}
```

6.
```
for(i = 0; i < n; i ++) {
        for (j = 0; j < n; j++) {
                for (k = 0; k < n; k++) {
                        x++;
                }
        }
}
```

7.
```
for(i = 1; i < n; i *= 2) {
        for (j = 0; j < n; j++) {
                x++;
                if (x < y)
                        x = 4;
        }
}
for(i = 0; i < n; i++)
        y = y * 3;
```