ALGORITHMS TUTORIAL 1
(Complexity and Order)
Date: Sep 5 – September – 2020

1. Put the following functions in order from lowest to highest in terms of their θ classes. (Some of the functions may be in the same θ class. Indicate that on your list also.)

   (a) $f_1(n) = n!$

   (b) $f_2(n) = n^{\frac{3}{2}}$

   (c) $f_3(n) = 1000$

   (d) $f_4(n) = \sqrt{n}(n + \log(n))$

   (e) $f_5(n) = 3^n$

   (f) $f_6(n) = 2^{(n+2)}$

   (g) $f_7(n) = 0.00001n$

2. Prove whether or not each of the following statements are true. For those that you believe are false, prove this by giving a counterexample (i.e. particular functions for f(n) and g(n) for which the given statement is not true). For those that you believe are true, use the formal definitions of big-oh, big-Ω, and big-Θ to prove it. In all problems, you are given that for all n, $f(n) \geqslant 0$ and $g(n) \geqslant 0$.

   (a) If $f(n) = O(g(n))$ then $g(n) = O(f(n))$

   (b) $\min(f(n), g(n)) = O(f(n) + g(n))$

   (c) If $f(n) = \Omega(g(n))$ then $g(n) = O(f(n))$

3. Give an exact solution to the following recurrences. Then use induction to prove that your solution is correct.

   (a) $T(n) = 3T(n/2) + n^2, T(1) = 1$ for $n \geqslant 0$ a power of 2

4. What is the asymptotic upper bound on the time complexity for the following code fragment? (For large values on n and assume that n is of the form 2^{2^k})

```
count = 1; //for n>=2
while(n>2)
{
    if(count % 2 == 1)
    {
        a++;
        n = pow(n, 1.2);
    }
    else
    {
        b++;
        n = pow(n, 0.4166667);
```

```
        }
        count++;
    }
```

5. Give an asymptotically tight solution to the recurrence.
   $T(n) = 9T(n/3) + n$

6. Propose an algorithm for computing 3^n using only $\Theta(\log(n))$ instructions. Show that your algorithm actually runs in $\Theta(\log(n))$ time.

7. Give a pseudo code for and matrix multiplication algorithm. How many instructions are executed when we multiply $n \times m$ matrix A with $m \times r$ matrix B ?

   Ans. m*n*r

8. Give an example of two positive real valued functions $f(n)$ and $g(n)$ of natural numbers that satisfy the property that $f(n)$ is not $O(g(n))$ and $g(n)$ is also not $O(f(n))$.

9. Write down the recurrence relation for the running time T(n) of the following code. What is the time complexity in big-Θ notation?

```
float useless(A){
    n = A.length;
    if (n==1){
        return A[0]; } // let A1,A2 be arrays of size n/2
    for (i=0; i <= (n/2)-1; i++){
        A1[i] = A[i];
        A2[i] = A[n/2 + i];
    }
    for (i=0; i<=(n/2)-1; i++){
        for (j=i+1; j<=(n/2)-1; j++){
            if (A1[i] == A2[j])
            A2[j] = 0;
    } }
    b1 = useless(A1);
    b2 = useless(A2);
    return max(b1,b2);
}
```