# (Sol) HW00 Time complexity solution

---

Give $\Theta$ running times for each of the following functions, as a function of their input parameter. HINT– if you are having trouble with finding $\Theta$ first find $O$, then $\Omega$.

Problem 1. (a) $\Theta(n^2)$. The outer loop runs $n$ times, and the inner loops run $3n$ times in total, so the overall time complexity is $\Theta(n^2)$.

```
public static int f1(int n){
  int sum = 0;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++)
      sum++;
    for (int j = 0; j < n; j++)
      sum++;
    for (int j = 0; j < n; j++)
      sum++;
  }
  return sum;
}
```

Problem 1. (b) $\Theta(n^2)$. $T_{f2}(n) \in \Theta(T_{f1}(n) + T_{f1}(n) + T_{f1}(n/2))$ = $\Theta(n^2 + n^2 + n^2/4)$ = $\Theta(n^2)$.

```
public static int f2(int n){
  int sum = 0;
  sum += f1(n) * f1(n);//run f1(n) + run f1(n) + add
  sum += f1(n/2);
  return sum;
}
```

For the following functions,

1) describe what this function computes, and

2) find $\Theta$ running time of the function

Problem 2.(a) `recursive1` returns $n$, for $n \geq 0$ in $n$ steps. `recursive2` returns the sum of $0, 1, 2, \cdots, n$, for $n \geq 0$. Its time complexity $T(n) = c + T(n-1) +$

$\Theta(recursive1(n))$. $\Theta(recursive1(n))$ is $\Theta(n)$, so $T(n) = T(n-1) + c + c'n$. Using recursion tree or substitution method, $T(n) \in \ \Theta(n^2)$

```
int recursive2(int n){
    if (n == 0)
        return 0;
    return recursive1(n) + recursive2(n-1);
}
```

```
int recursive1(int n){
    if (n == 0)
        return 0;
    else
        return 1 + recursive1(n-1);
}
```

Problem 2.(b) This function returns the sum of $1 + 1 + 2 + \cdots + 2^{n-1}$, so $2^n$, for $n \geq 0$. Every increment is a constant amount of work, and the number of increments `recursive3` does doubles every time $n$ is increased by 1, so $\Theta(2^n)$.

```
int recursive3(int n){
  if (n == 0)
    return 1;
  else
    return recursive3(n-1) +  recursive3(n-1);
}
```

Problem 2.(c) This function returns the same value as $recursive3$, so $2^n$, for $n \geq 0$. However, the amount of work is increased by one multiplication every time $n$ is increased by 1, so $\Theta(n)$

```
int recursive4(int n){
  if (n == 0)
    return 1;
  else
    return 2 * recursive4(n-1);
}
```

Problem 2.(d) (assume $n$ is a power of 2, e.g. $n = 16$) This function returns $n$ for $n \geq 1$, and $1$ for $n < 1$. Since the number of function calls is $1 + 2 + 4 + \cdots + n = \Theta(n \lg n)$

```
int recursive5(int n){
  if (n <= 1)
    return  n;
  int dummy = 0;
  for (int i = 0; i < n; i++)
    dummy++;
  if (n % 2 != 0)
    return 1 + recursive5(n-1);
  return recursive5(n/2) + recursive5(n/2);
}
```