Lab 2

1. Big O Notation

1. Let $f(n) = 2n + 6$. Is $f(n) = O(n)$
2. Let $f(n) = 3n^2 + 4n - 8$. Is $f(n) = O(n^2)$
3. Prove or disprove that $log(n!)$ is $O(nlogn)$
4. Prove that $2n + 3$ is $O(n^2)$
5. Prove that 2^{n+2} is $O(2^n)$

2. Write the recurrence relation of the following algorithms

1.

$$T(n) = T(n-1) + 4$$
$$T(0) = 1$$

2.

$$T(n) = T(n-1) + 8$$
$$T(1) = 8$$

3.

$$T(n) = 4T(\frac{n}{2}) + n$$

4.

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2$$

5.

$$T(n) = 2T\left(\frac{n}{8}\right) + \sqrt[3]{n}$$

6.

```
int factorial(int n){
    if (n==1)
        return 1;
    return n * factorial(n-1);
}
```

Find time complexity of the following code

1.

```
void foo(int n){
     int i = 1; int s = 1;
     while (s <= n){
         i++;
         s = s + i;
         System.out.print("*");
     }
}
```

2.

```
void foo(int n){
     int count = 0;
    for (int i = n / 2; i <= n ; i++) {
        for(int j = 1; j + (n/2) <= n; j++){
            for(int k = 1; k <= n; k*=2){
                count++;
            }
        }
    }
}
```

3.

```
void foo(int n){
     int count = 0;
    for (int i = n / 2; i <= n ; i++) {
        for(int j = 1; j <= n; j*=2){
            for(int k = 1; k <= n; k*=2){
                count++;
            }
        }
    }
}
```

4.

```
void foo(int n){
    if(n == 1) return;
    for (int i = 1; i <= n ; i++) {
        for(int j = 1; j <= n; j++){
            System.out.print("*");
            break;
        }
    }
}
```

5.

```
void foo(int n){
    int a = 0; int i = n;
    while (i > 0){
        a += i;
        i /= 2;
    }
}
```

6.

```
void foo(int n){
     int count = 0;
    for(int i = n; i > 0; i /= 2){
        for(int j = 0; j < i; j++){
            count++;
        }
    }
}
```

7.

```
void foo(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i; j < i * i; j++) {
            if (j % i == 0) {
                for (int k = 0; k < j; k++) {
                    System.out.print("*");
                }
            }
        }
    }
}
```