# TIME COMPLEXITY QUESTIONS

**Question :** Find the Time Complexity of the following:

a)

```
int i, j, k = 0;
        for (i = n / 2; i <= n; i++) {
            for (j = 2; j <= n; j = j * 2) {
                k = k + n / 2;
            }
        }
```

A. O(n)
B. O(N log N)
C. O(n^2)
D. O(n^2Logn)

b)

```
for(int i=0;i<n;i++)
          i*=k
```

Here, k is some constant value

A. O(n)
B. O(k)
C. O(logkn) (= logn of base k)
D. O(lognk) (= logk of base n)

c)

Algorithm A and B have a worst-case running time of O(n) and O(logn), respectively. Therefore, algorithm B always runs faster than algorithm A.

A. True
B. False

d) Find the time & space complexity of floorSqrt function in the following code to calculate square root of a number :

```java
class SqrtNum {

   static int floorSqrt(int x)
   {
       if (x == 0 || x == 1)
           return x;

       int i = 1, result = 1;

       while (result <= x) {
           i++;
           result = i * i;
       }
       return i - 1;
   }

   public static void main(String[] args)
   {
       int x = 11;
       System.out.print(floorSqrt(x));
   }
}
```

e) Find the time & space complexity of the following code:

```java
int a = 0;
for (int i = 0; i < n; ++i) {
   for (int j = n; j > i; --j) {
       a = a + i + j;
   }
}
```