1.1

Determine how many times the output statement is displayed in each of the following fragments. Indicate whether the algorithm is **O**(n) or **O**(n^2).

a.
```
for (int i = 0; i < n; i++)
    for (int j = 0; j < n; j++)
        System.out.println(i + " " + j);
```

b.
```
for (int i = 0; i < n; i++)
    for (int j = 0; j < 2; j++)
        System.out.println(i + " " + j);
```

c.
```
for (int i = 0; i < n; i++)
    for (int j = n - 1; j >= i; j--)
        System.out.println(i + " " + j);
```

d.
```
for (int i = 1; i < n; i++)
    for (int j = 0; j < i; j++)
        if (j % i == 0)
            System.out.println(i + " " + j);
```

a. n^2 $O(n^2)$

b. $2n$ $O(n)$

c. $n(n-1)/2$ $O(n^2)$

d. $n-1$ $O(n)$

1.3

How does the performance grow as n goes from 2000 to 4000 for the following? Answer the same question as n goes from 4000 to 8000. Provide tables similar to Table 2.4.

a. **O**(log n)
b. **O**(n)
c. **O**(n log n)
d. **O**(n^2)
e. **O**(n3)

$O(f(n))$	$f(2000)$	$f(4000)$	$f(8000)/f(4000)$
$O(\log n)$	10.97	11.97	1.09
$O(n)$	2000	4000	2
$O(n \log n)$	21932	47863	2.18
$O(n^2)$	4000000	16000000	4
$O(n3)$	8×10^9	6.4×10^{10}	8

$O(f(n))$	$f(4000)$	$f(8000)$	$f(100)/f(50)$
$O(\log n)$	11.97	12.97	1.08
$O(n)$	4000	8000	2
$O(n \log n)$	47863	103726	2.17
$O(n^2)$	16000000	64000000	4
$O(n3)$	6.4×10^{10}	5.12×10^{11}	8

2.1