Exercise #10

```
int sum = 0;
    for (int counter = n; counter >0; counter = counter -2)
            sum = sum + counter;
```

if n = 8

then n = 8

1. counter = 8
   counter = 8 - 2
   Counter = 6

2. counter = 6
   counter = 6 - 2
   counter = 4

3. counter = 4
   counter = 4 - 2
   counter = 2

4. counter = 2
   counter = 2 - 2
   counter = 0

It will execute 4 times which is $\frac{n}{2}$ times.

$\frac{n}{2} = O(n)$

Exercise #12

```
for (int pass = 1; pass <= n; pass++){
        for (int i = 0; i < n; i++) {
            for (int count = 1; count < 10; count++){
                ...
            }
        }
}
```

for (int pass = 1; pass <= n; pass++) $= n$

for (int i = 0; i < n; i++) $= n$

for (int count = 1; count < 10; count++) $= 1$

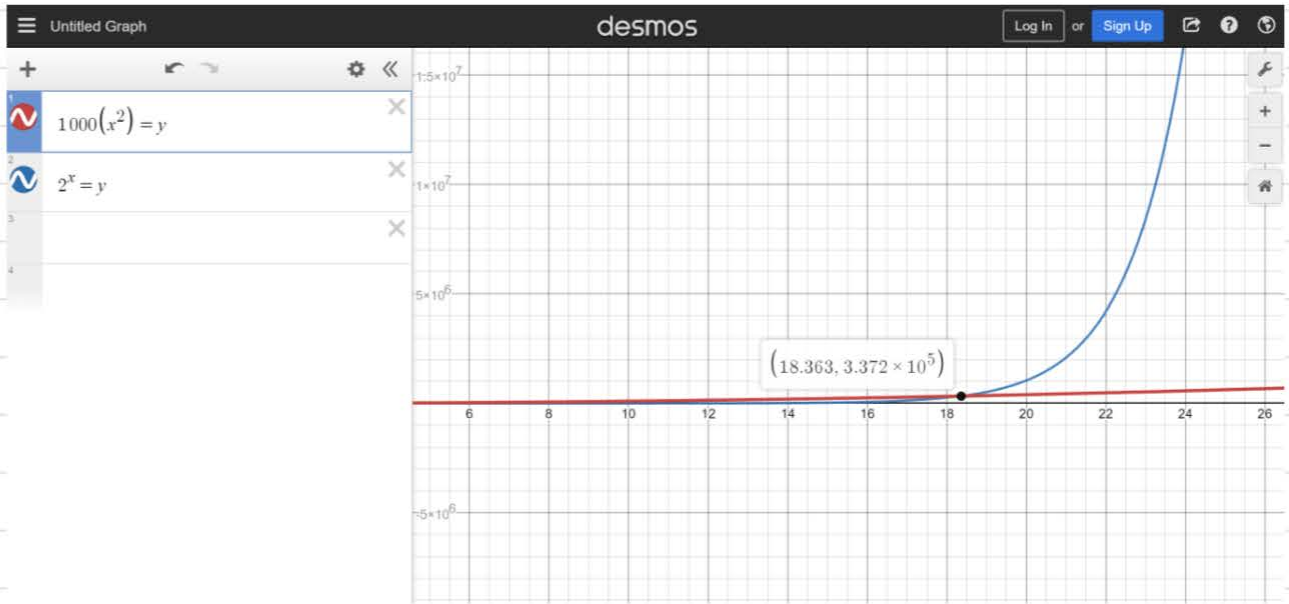
$n(n(1)) = n(n) = n^2$

$O(n^2)$

Exercise #16

Program A requires $1000n^2$ operations.
Program B requires 2^n operations.
What values of n will A execute faster than B?

	A	B
0	0	1
1	1000	2
2	4000	4
10	100 000	1024
11	121 000	2048
12	144 000	4096
20	400 000	1048076
17	289 000	131 072
18	324 000	262 144
19	361 000	542 288



Program A will execute faster than Program B when $n >= 19$.

a more accurate number would be when $n >= 18.363$

Exercise #24a

write java code for a $O(n^2)$ program that will find an integer not in an array.

```java

public class exercise24a {
    public static int findMissing (int[] arr){
        int n = arr.length;

        for (int counter = 1; counter <= n + 1; counter++) {
            for (int i = 0; i < n; i++) {
                if (arr[i] == counter) {
                    break;
                }
                if (i == n-1) {
                    return counter;
                }
            }
        }
    }
}

```