Monday Dec. 12

Exam Review 2

```
int nob(int rows) {
  if(rows == 1) {
    return 1;
  }
  else {
    return rows + nob(rows - 1);
  }
}
```

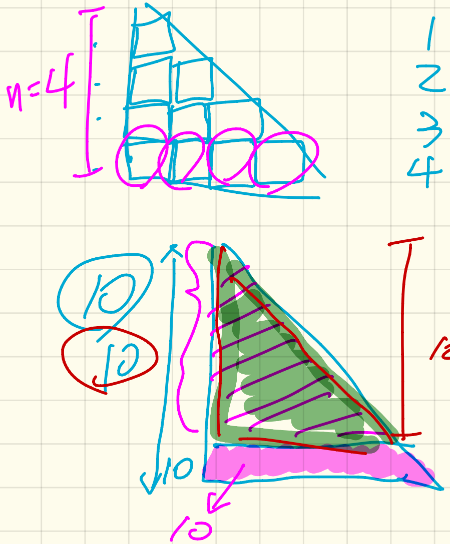
nob(10)

10

9

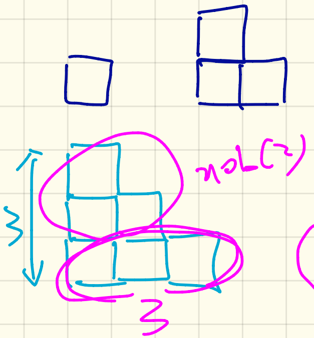
nob (10-1)

4.



1
2
3
4

10



6

nob (3)

3 + nob(2)

2 + nob(1)

3

1

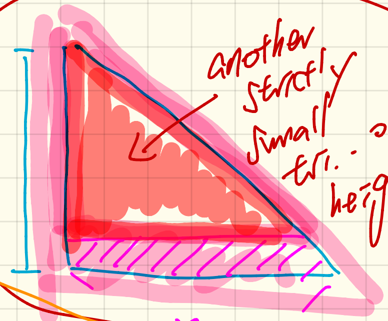
```
int nob(int rows) {
1   if(rows == 1) {
2     return 1;
    }
    else {
3     return rows + nob(rows - 1);
    }
}
```

Prove: nob(n), $n > 0$, returns the numb. of blocks of a triangle of height n.

Proof. 1. Base case ("triangle" of height 1)

(concept) □ 1 <1 returns 1

2. Recursive case (triangle of height ~~'rows'~~ n)



I.H. nob($n-1$) returns the num. of blocks of a triangle of height $n-1$

By I.H. we know nob($n-1$) returns the expected result.

To compute n.o.b. of a triangle of height n, we have:

n + n.o.b. of tri. of h. $n-1$

<3 I.H. <3 nob($n-1$)

```
int nob(int rows) {
   if(rows == 1) {
      return 1;
   }
   else {
      return rows + nob(rows - 1);
   }
}
```

R.T. ?   T(?)

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

for some constant c

$$T(1) = 1$$

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

$$= (T(n-2) + 1) + 1$$

$$= ((T(n-3) + 1) + 1) + 1$$

⋮

$$= ((T(1) + 1) + 1 \cdots + 1$$

$n-(n-1)$ $n-1$

$(n-1)\cdot c$

$$= n-1$$

$O(n)$

```
class Counter1 {
    int v;
    void inc() {
        v++ };
}
```

non-static

each instance has its own copy

```
class Counter2 {
    static int v;
    void inc() {
        v++
    }
}
```

all instances share the same copy

Counter2

v | 0

c3 → Cou2

c4 → Cou2

```
Counter1 c1 = new Counter1();
Counter1 c2 = new Counter1();
print (c1.value); 0
print (c2.value); 0
c1.inc();
print (c1.value); 1
print (c2.value); 0
```

c1 → Cou. v | 0 1

c2 → Cou v | 0

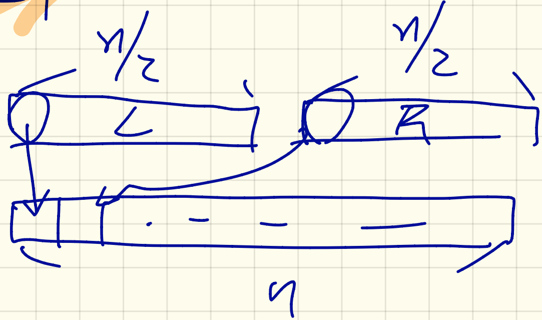
```
Counter2 c3 = new C2();
Counter2 c4 = new C2();
print (c3.value); 0
print (c4.value); 0
c3.inc();
print (c3.value);
print (c4.value);
```

Counter2.value

if (t.size() <= 1) {
return t
}

$T(0) = 1$

$T(1) = 1$



ms(sublist(n/2)) $T(\frac{n}{2})$
left

ms(sublist(n/2)) $T(\frac{n}{2})$
right

merge $O(n)$

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