Recurrences

```
int length(ListNode list)
{
  if (0 == list) return 0;
  else return 1+length(list.getNext());
}
```

T(n) = time to compute length for an n-node list

$T(0) = 0$

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

Give a guess for the “closed form solution” for T(n)

Solving recurrence relations by repeated substitution

```
T(1) = a
T(n) = T(n-1) + b

T(n) = T(n-1) + b
     = T(n-2) + b + b = T(n-2) + 2b
     = T(n-3) + 3b
     …
     = T(n-i) + ib
```

In order to get to the base case of T(1) what value of i do we want?

Solving recurrence relations by repeated substitution

```
T(1) = a
T(n) = T(n-1) + b

T(n) = T(n-1) + b
     = T(n-2) + b + b = T(n-2) + 2b
     = T(n-3) + 3b
     …
     = T(n-i) + ib
```

Given that an i of n-1 gives us the base case of T(1), what is the "closed form solution" of T(n)?

Checking our result by induction

```
T(1) = a
T(n) = T(n-1) + b

T(n) = a + (n-1)b ??
```

Does our possible closed form solution work for the base case? Why?

Checking our result by induction

```
T(1) = a
T(n) = T(n-1) + b

T(n) = a + (n-1)b ??
```

If our closed form solution works for T(n-1) then does our possible closed form solution give us the correct value for T(n)? Why?

Solving recurrence relations by repeated substitution

$$
\begin{aligned}
T(1) &= 1 \\
T(n) &= 2T(n/2) + 1 \\
T(n) &= 2T(n/2) + 1 \\
&= 2^2T(n/2^2) + 2 + 1 \\
&= 2^3T(n/2^3) + 2^2 + 2^1 + 2^0 \\
&\ldots \\
&= 2^iT(n/2^i) + 2^{i-1} + \ldots + 2^1 + 2^0
\end{aligned}
$$

In order to get to the base case of T(1) what value of i do we want?

Solving recurrence relations by repeated substitution

$T(1) = 1$

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

$T(n) = 2^iT(n/2^i) + 2^{i-1} + \ldots + 2^1 + 2^0$

Given that an i of $\log_2 n$ gives us the base case of T(1), what is the "closed form solution" of T(n)? Hint: Use a formula on p16

Solving recurrence relations by repeated substitution

```
T(1) = 1
T(n) = 2T(n/2) + 1

T(n) = 2n - 1
```

Check out the answer with induction!