Recursion

A recursive function is one that calls itself. Different invocations of the function have their own local variable scope, so this is safe. As an example, consider summing all the numbers below (and including) some input number:

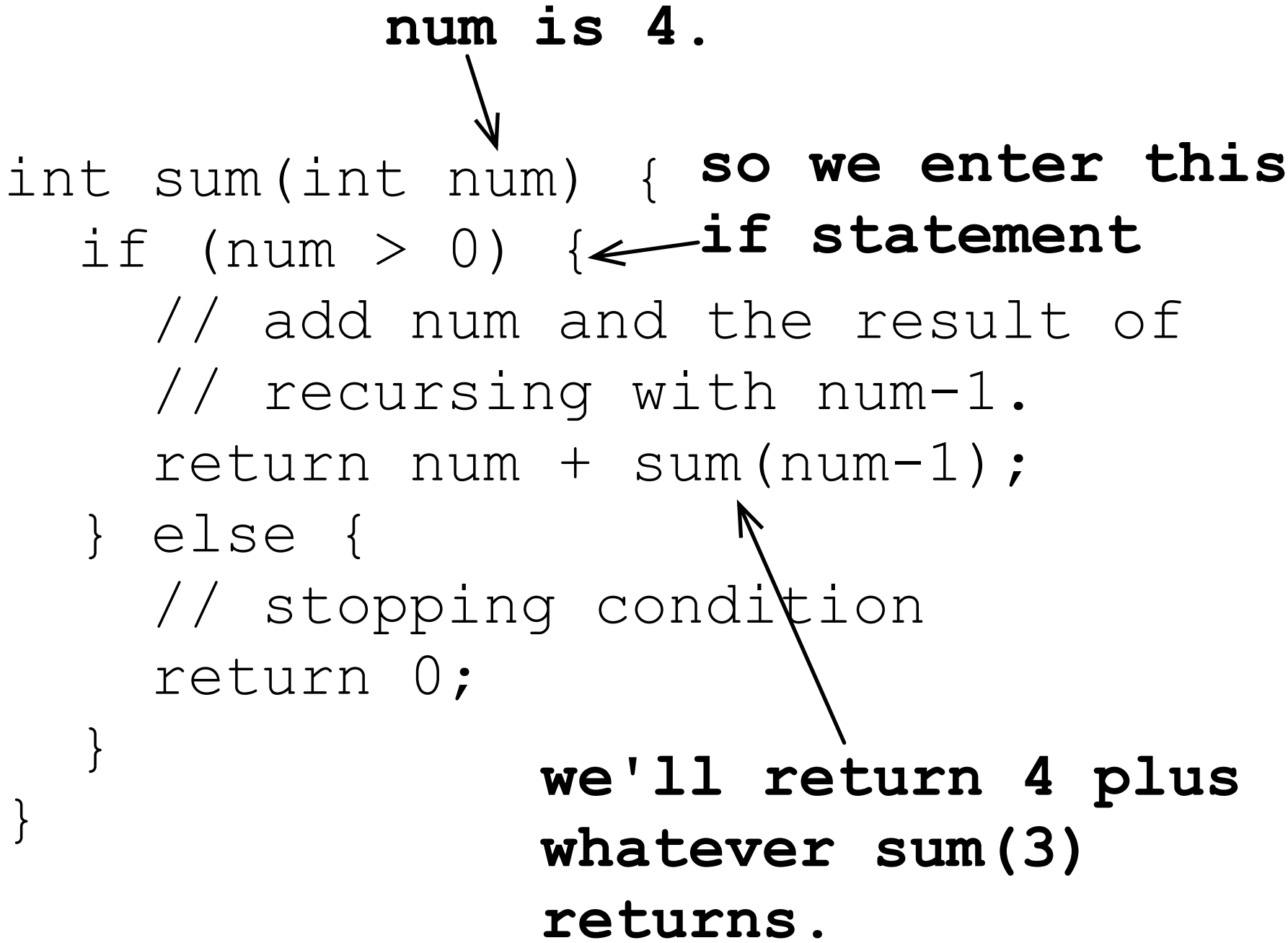
```
def add_it_up(num):
    if (num > 0):
        return num + add_it_up(num - 1)
    else:
        return 0
```

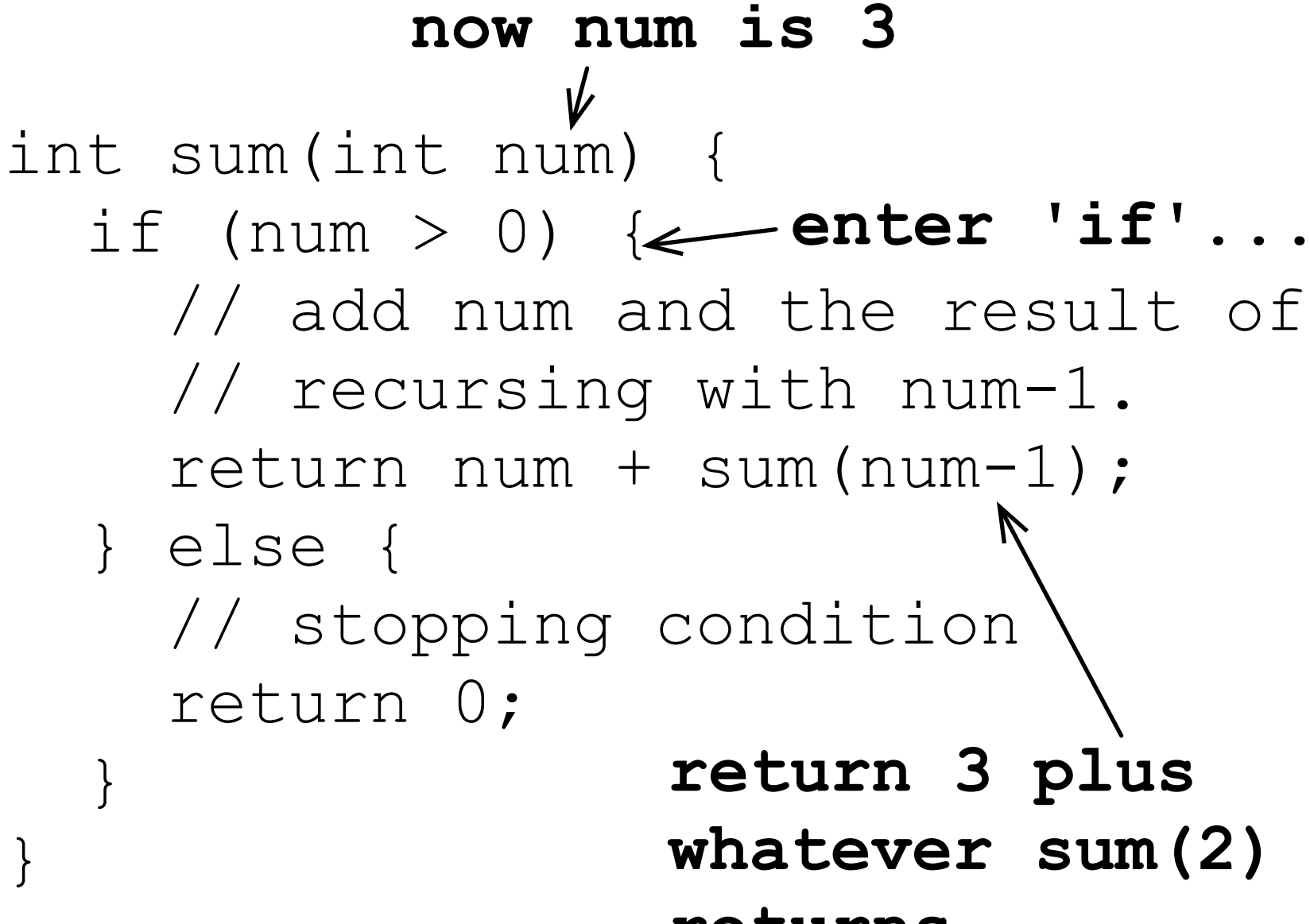
calling **add_it_up(6)** gives 21:

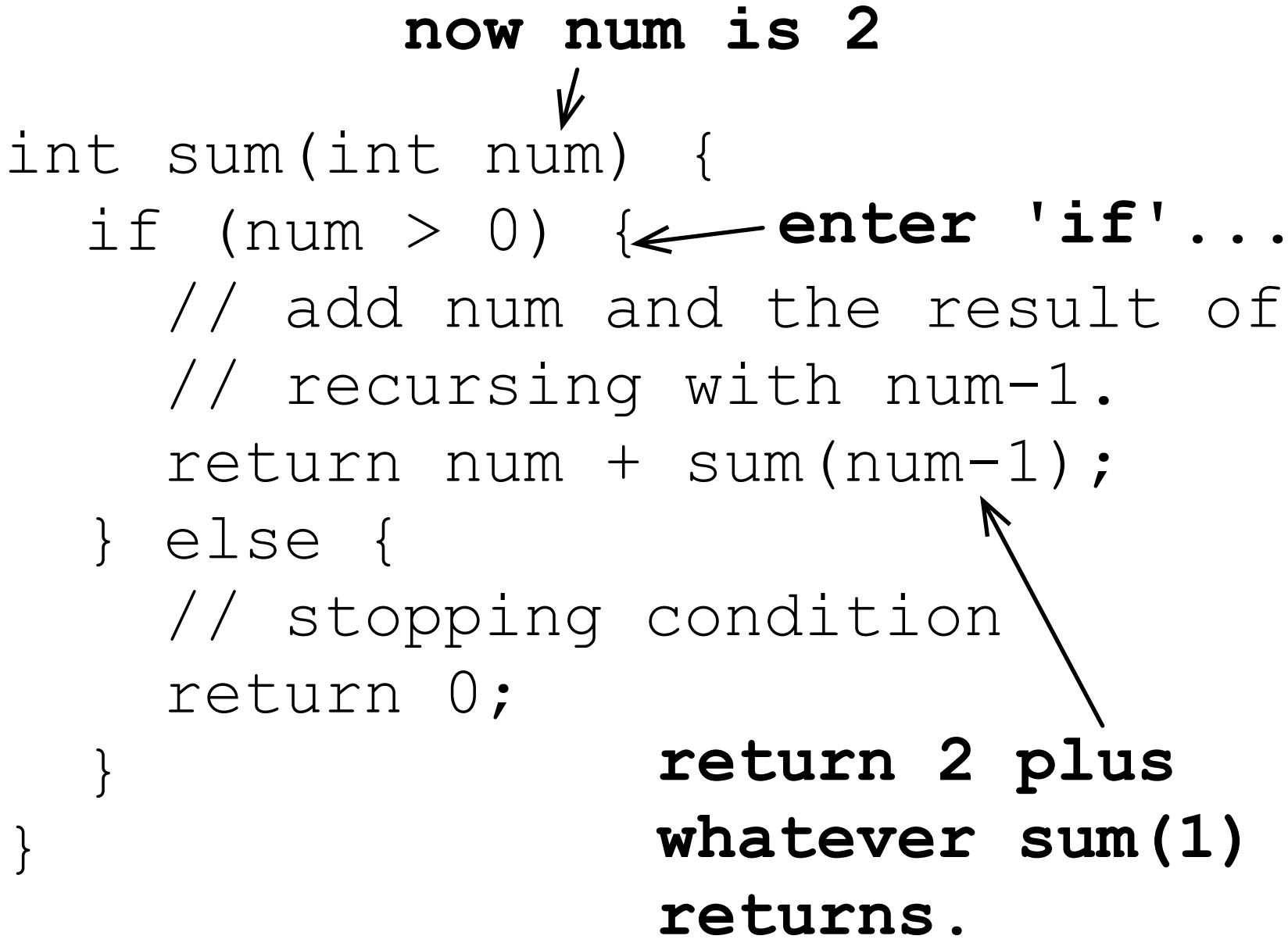
6 + 5 + 4 + 3 + 2 + 1 + 0

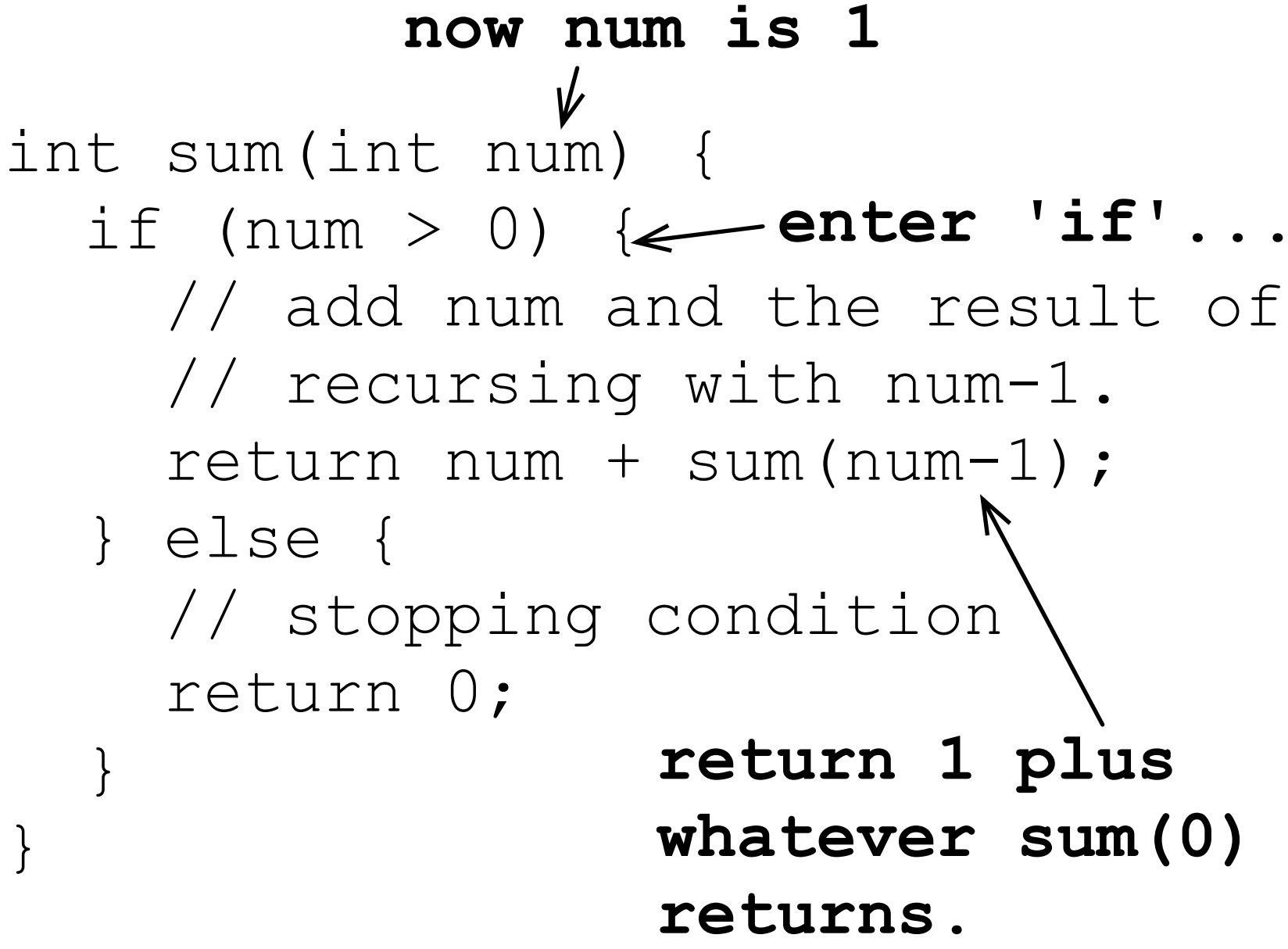
If we call sum(4), here's what happens.

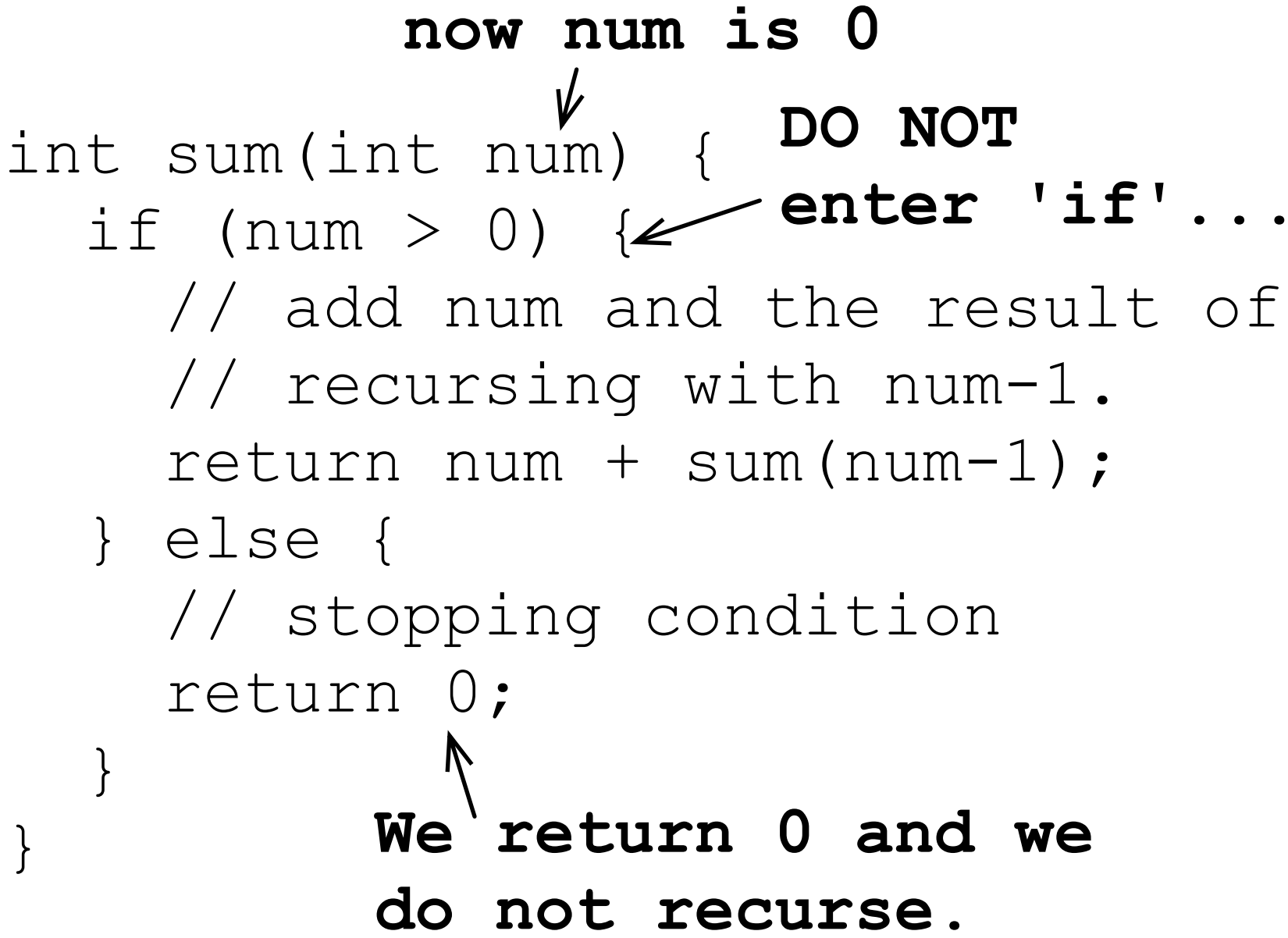
```
int sum(int num) {
  if (num > 0) {
    // add num and the result of
    // recursing with num-1.
    return num + sum(num-1);
  } else {
    // stopping condition
    return 0;
  }
}
```


num is 4.
int sum(int num) {
so we enter this
if (num > 0) {
if statement
// add num and the result of
// recursing with num-1.
return num + sum(num-1);
} else {
// stopping condition
return 0;
}
}
we'll return 4 plus
whatever sum(3)
returns.

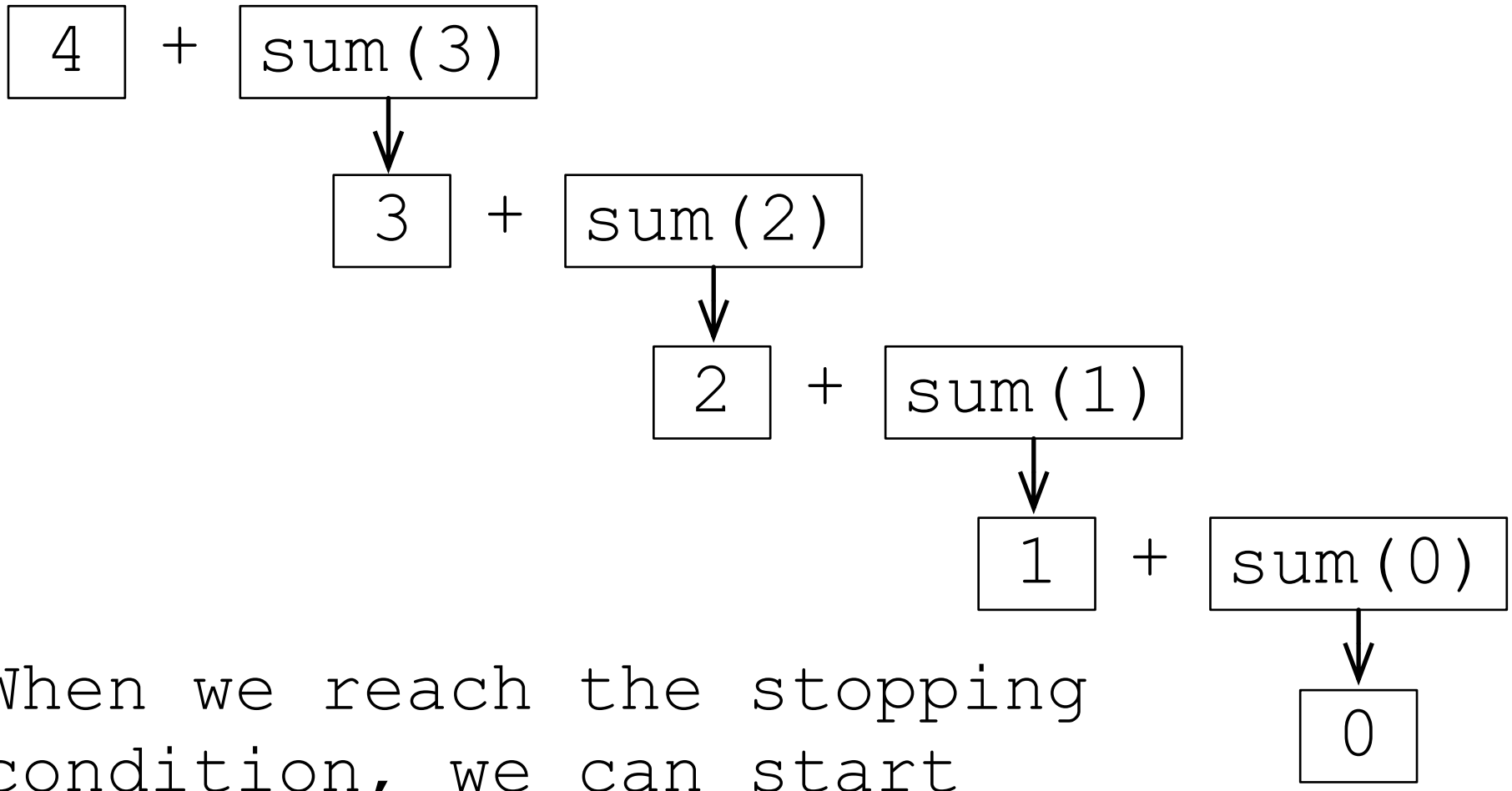

now num is 3
int sum(int num) {
if (num > 0) {
enter 'if'...
// add num and the result of
// recursing with num-1.
return num + sum(num-1);
} else {
// stopping condition
return 0;
}
}
return 3 plus
whatever sum(2)
returns.


now num is 2
int sum(int num) {
if (num > 0) {
enter 'if'...
// add num and the result of
// recursing with num-1.
return num + sum(num-1);
} else {
// stopping condition
return 0;
}
}
return 2 plus
whatever sum(1)
returns.

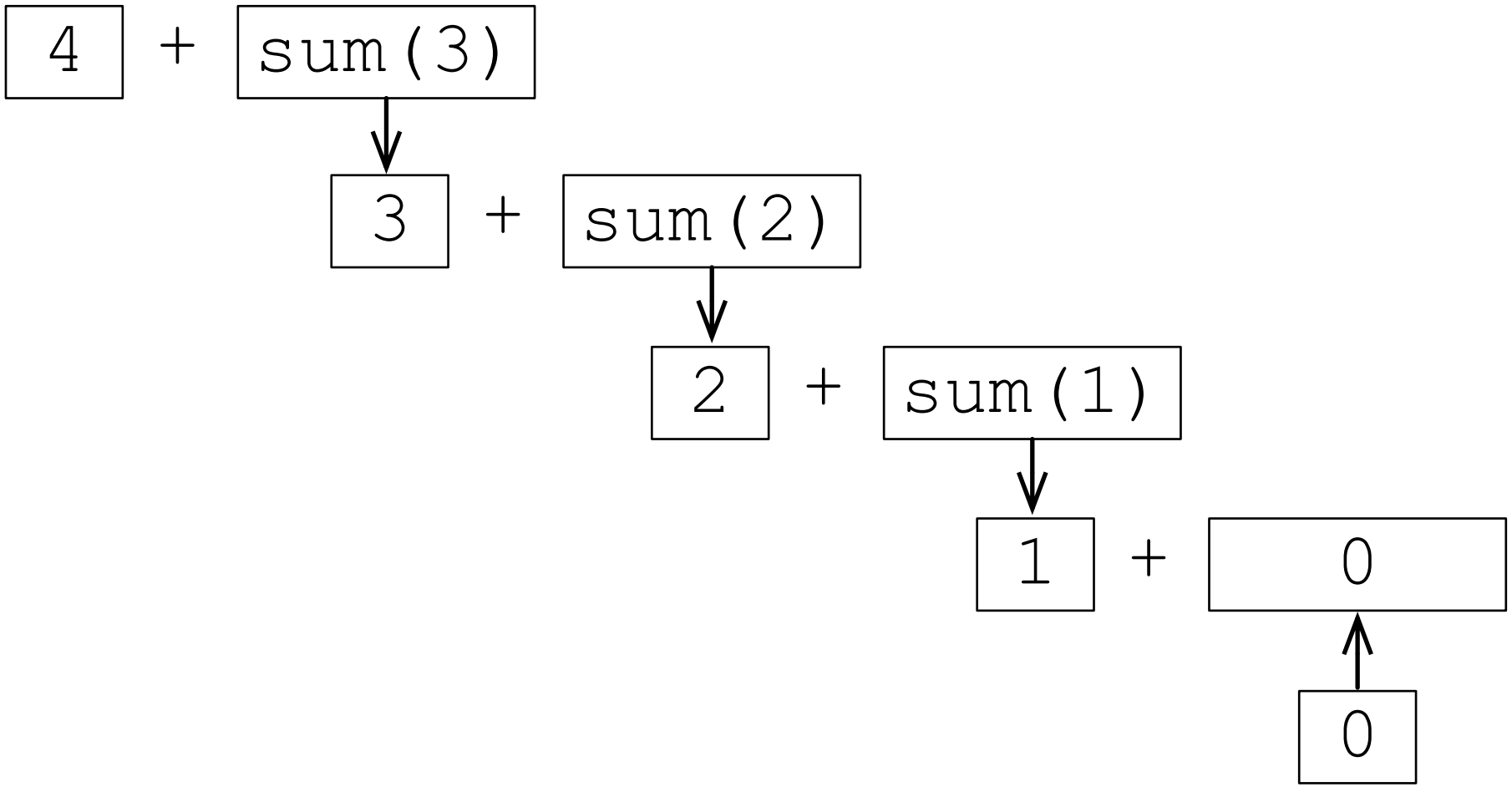

now num is 1
int sum(int num) {
if (num > 0) {
enter 'if'...
// add num and the result of
// recursing with num-1.
return num + sum(num-1);
} else {
// stopping condition
return 0;
}
}
return 1 plus
whatever sum(0)
returns.

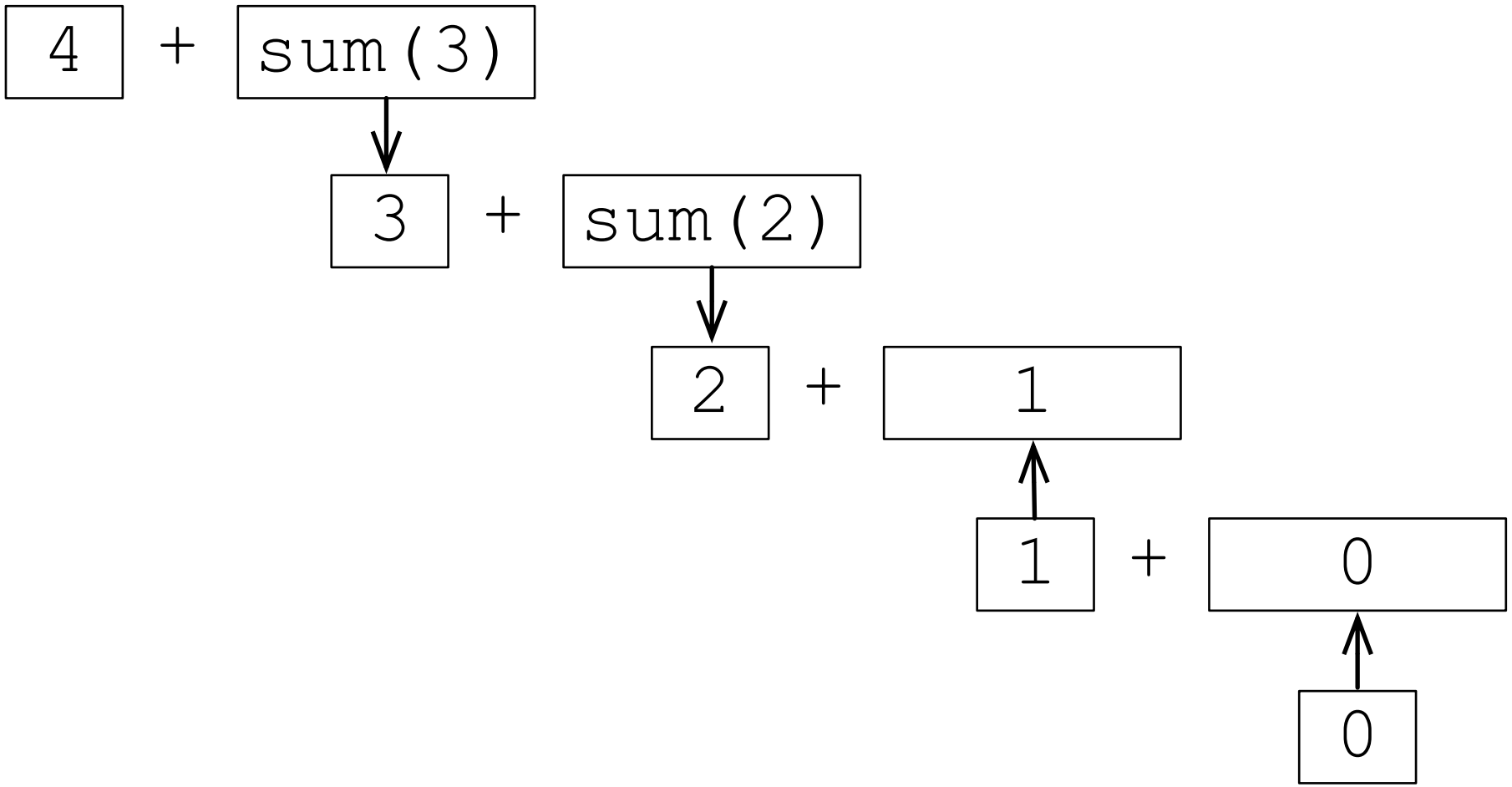

now num is 0
DO NOT
enter 'if'...
int sum(int num) {
if (num > 0) {
// add num and the result of
// recursing with num-1.
return num + sum(num-1);
} else {
// stopping condition
return 0;
}
}
We return 0 and we
do not recurse.

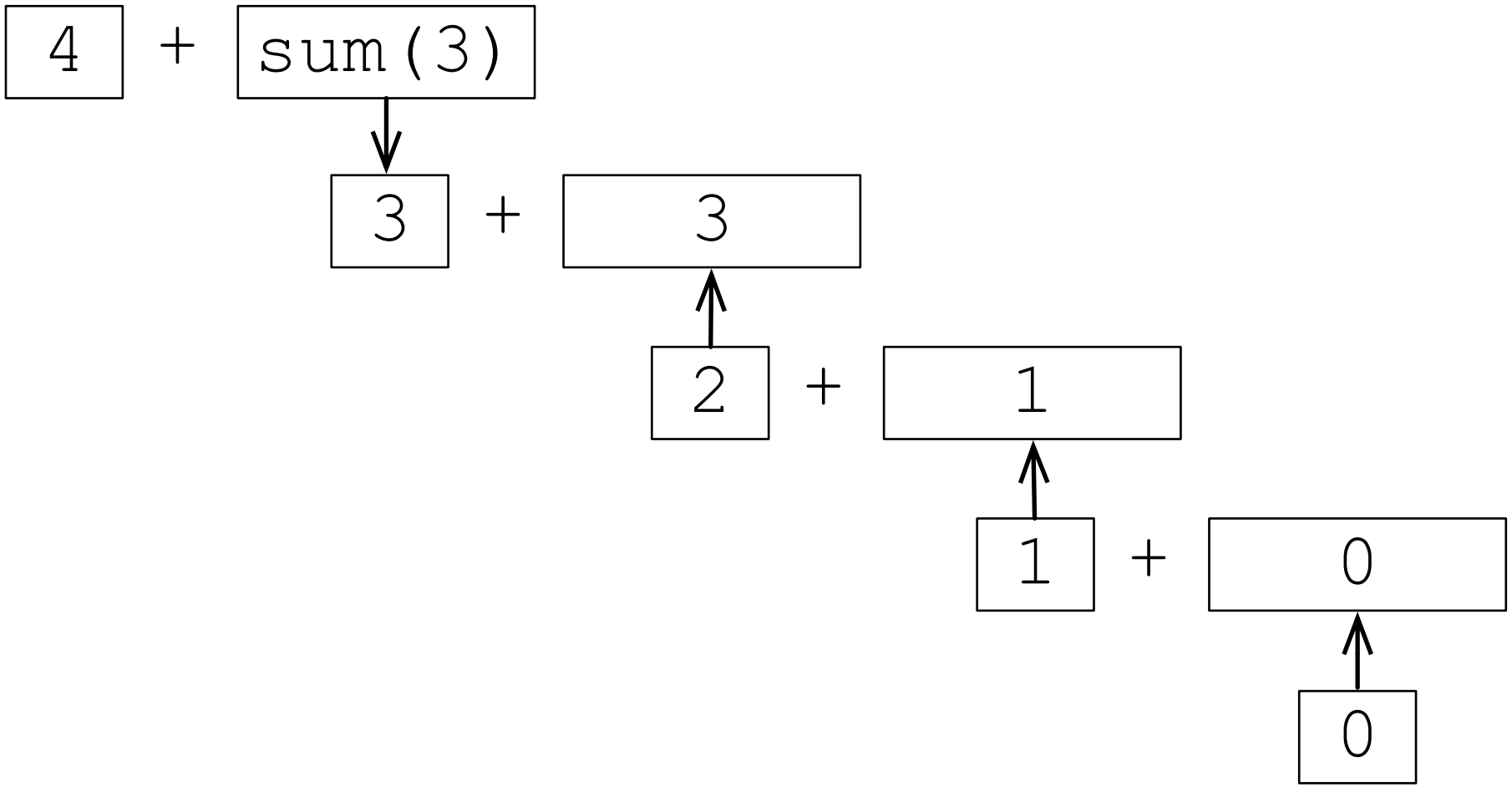
That recursive call stack looks like this:

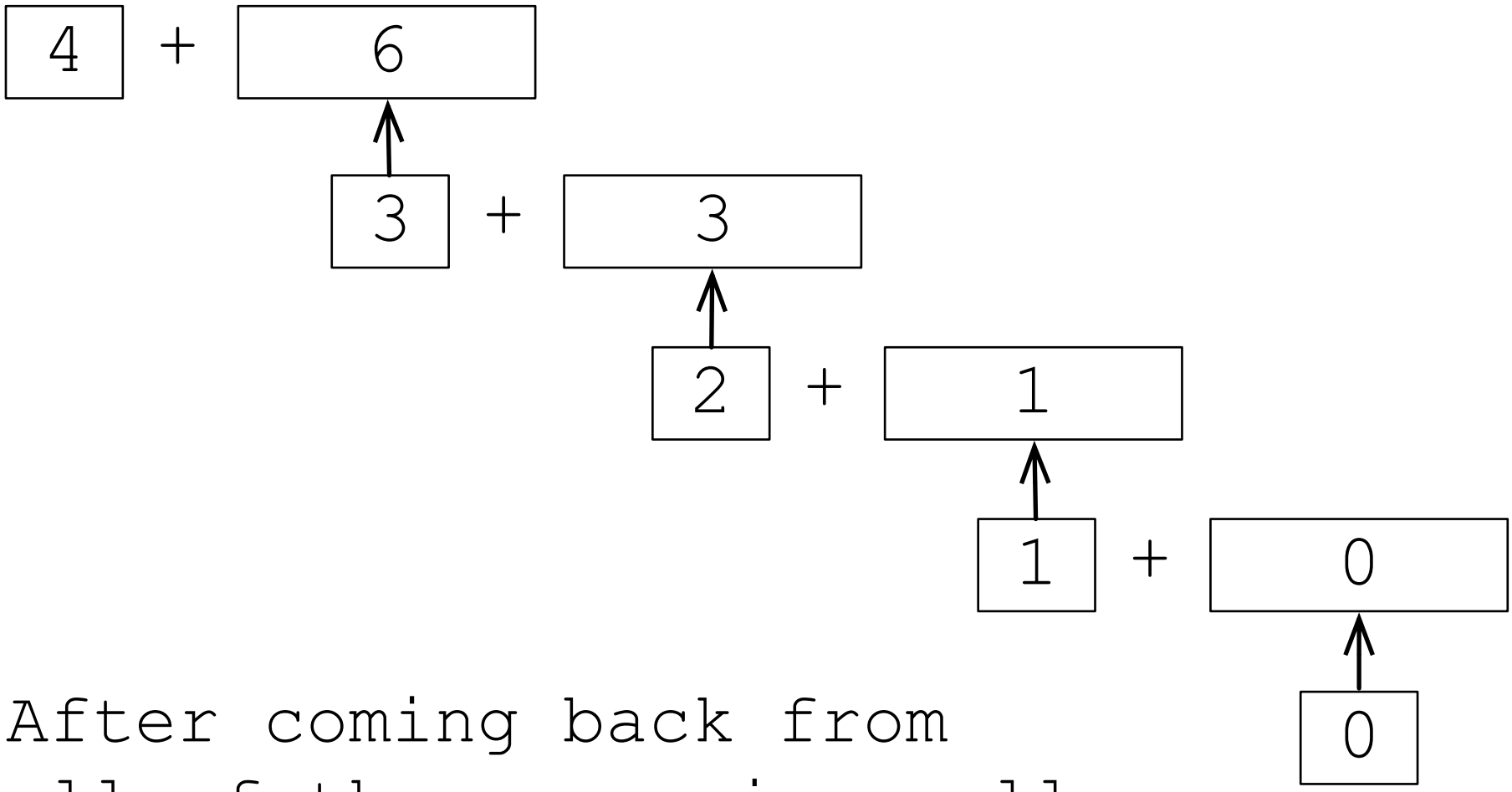


When we reach the stopping condition, we can start filling in the sum() calls with the proper return value.


4
+
sum(3)
3
+
sum(2)
2
+
sum(1)
1
+
0
0


4
+
sum(3)
3
+
sum(2)
2
+
1
1
+
0
0


4
+
sum(3)
3
+
3
2
+
1
1
+
0
0



After coming back from
all of the recursive calls,
the result is 4 + 6 = **10.**

In any recursive function, here are some things to keep in mind:

1. The function will either recurse, or it won't.
2. What's the stopping condition? When should it *not* recurse?
3. If it does recurse, how does the recursion contribute to the overall goal? (E.g. how do you use the result?)
4. Are there conditions for which it will recurse forever? (If so, you should probably detect those conditions and avoid recursing.)