Recursion

Recursive Functions

- **What is recursion?**

  When one function calls ITSELF directly or indirectly.
- A function is a ***recursive*** if a statement in the body of the function calls the function that contains it.

Recursive Functions

Why learn recursion?

- New mode of thinking.
- Powerful programming tool.
- Divide-and-conquer paradigm.
- Many computations are naturally self- referential

Recursive Functions (cont.)

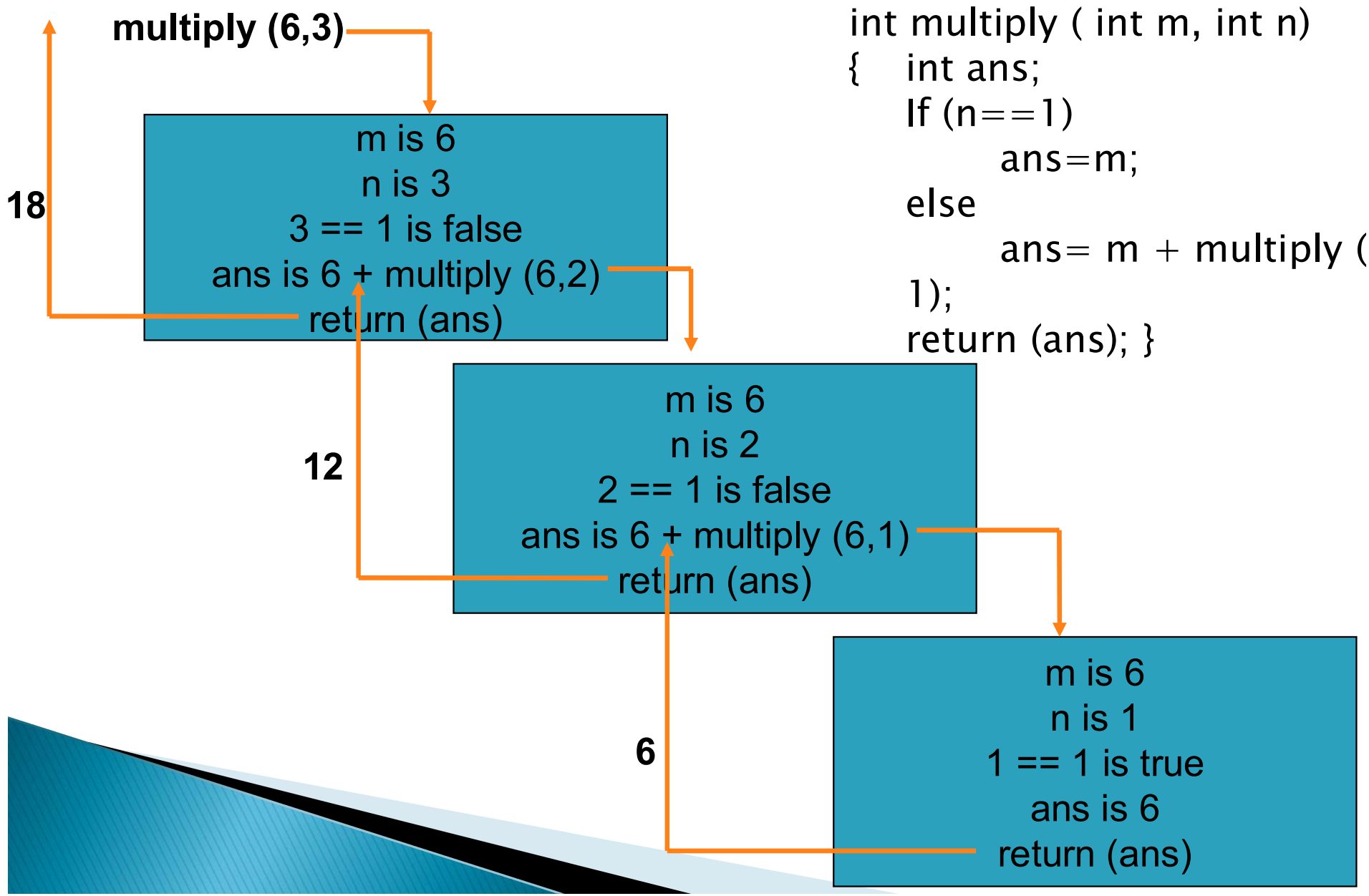
- A recursive definition is made up of two parts.
  - Base case that tells us directly what the answer is.
  - Recursive case that defines the answer in terms of the answer to a subproblem.
- For example, in factorial:
  - Base case is factorial(0)=1.
  - Recursive case is factorial(n) = n * factorial(n-1).

Recursive Functions

- ***Recursive Function Multiply***

```
int multiply ( int m, int n)
{
int ans;
If (n==1)
   ans=m;
else
   ans= m + multiply (m, n-1);
return (ans);
}
```

Recursive Functions




```
int multiply ( int m, int n)
{   int ans;
    If (n==1)
            ans=m;
    else
            ans= m + multiply (m, n-
    1);
    return (ans); }
```

Recursive Functions

- When a function calls itself, new local variables are allocated storage on the stack, and the function code is executed with these new variables from the beginning .
- A recursive call does not make a new copy of the function. Only the arguments are new.

Recursive Functions – Example

```
#include <stdio.h>
int sum(int n)
{   if(n==0)
            return n;
    else return n+sum(n-1);
}
int main()
{   int num,add;
    printf("Enter a positive integer:\n");
    scanf("%d",&num);
    add=sum(num);
    printf("sum=%d",add);
}
```

Recursive Functions – Example

```
#include <stdio.h>
int sum(int n);
int main()
{ int num,add;
  printf("Enter a positive integer:\n");
  scanf("%d",&num);
  add=sum(num);
  printf("sum=%d",add);
}

int sum(int n)
{ if(n==0)
    return n;
else return n+sum(n-1);
}
```

$$
\begin{aligned}
&\text{sum}(5)\\
&=5+\text{sum}(4)\\
&=5+4+\text{sum}(3)\\
&=5+4+3+\text{sum}(2)\\
&=5+4+3+2+\text{sum}(1)\\
&=5+4+3+2+1+\text{sum}(0)\\
&=5+4+3+2+1+0\\
&=5+4+3+2+1\\
&=5+4+3+3\\
&=5+4+6\\
&=5+10\\
&=15
\end{aligned}
$$

Recursive Functions – Factorial

- Functions are very often defined recursively. The classic example is the factorial function.
  - factorial(0) = 1
  - factorial(n) = n * factorial(n-1) [for n>0]
- Let's compute factorial(3).
  - factorial(3) = 3 * factorial(2)
    = 3 * 2 * factorial(1)
    = 3 * 2 * 1 * factorial(0)
    = 3 * 2 * 1 * 1
    = 6

Writing Recursive C Functions Factorial

- int factorial(int n)

```
{
    if (n == 0) return 1 ;
    else  return n * factorial(n-1) ;
}
```

Fibonnacci Series
0, 1, 1, 2, 3, 5, 8, 13, 21, …

- Base Cases:
  - fib(0) = 0, fib(1) = 1
- Recursive Case (two recursive calls):
  - fib(n) = fib(n-1) + fib(n-2) [for n>1]
- fib(4)= fib(3) + fib(2)

  = fib(2) + fib(1) + fib(1) + fib(0)

  = fib(1) + fib (0) + 1 + 1 + 0

  = 1 +0 + 1 + 1 + 0

  = 3

Writing Recursive C Functions Fibonacci

- int fib(int n)

```
{
    if (n == 0) return 0;
    if (n == 1) return 1;
    else return (fib(n-1) + fib(n-2));
}
```

Recursive Functions

- $F(X) = 2 * X + F(X - 2)$;
- $F(0) = F(1) = 0$;