Recursion

Week #6

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.

Example 1: Multiplication

- Suppose we wish to write a recursive function to multiply an integer m by another integer n using addition. [We can add, but we only know how to multiply by 1].
- The best way to go about this is to formulate the solution by identifying the base case and the recursive case.
- The base case is if n is 1. The answer is m.
- The recursive case is: m*n = m + m (n-1).

$$m*n \begin{cases} m, & n = 1 \\ m + m\,(n-1), & n>1 \end{cases}$$

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);
}
```

Example 1: Multiplication …

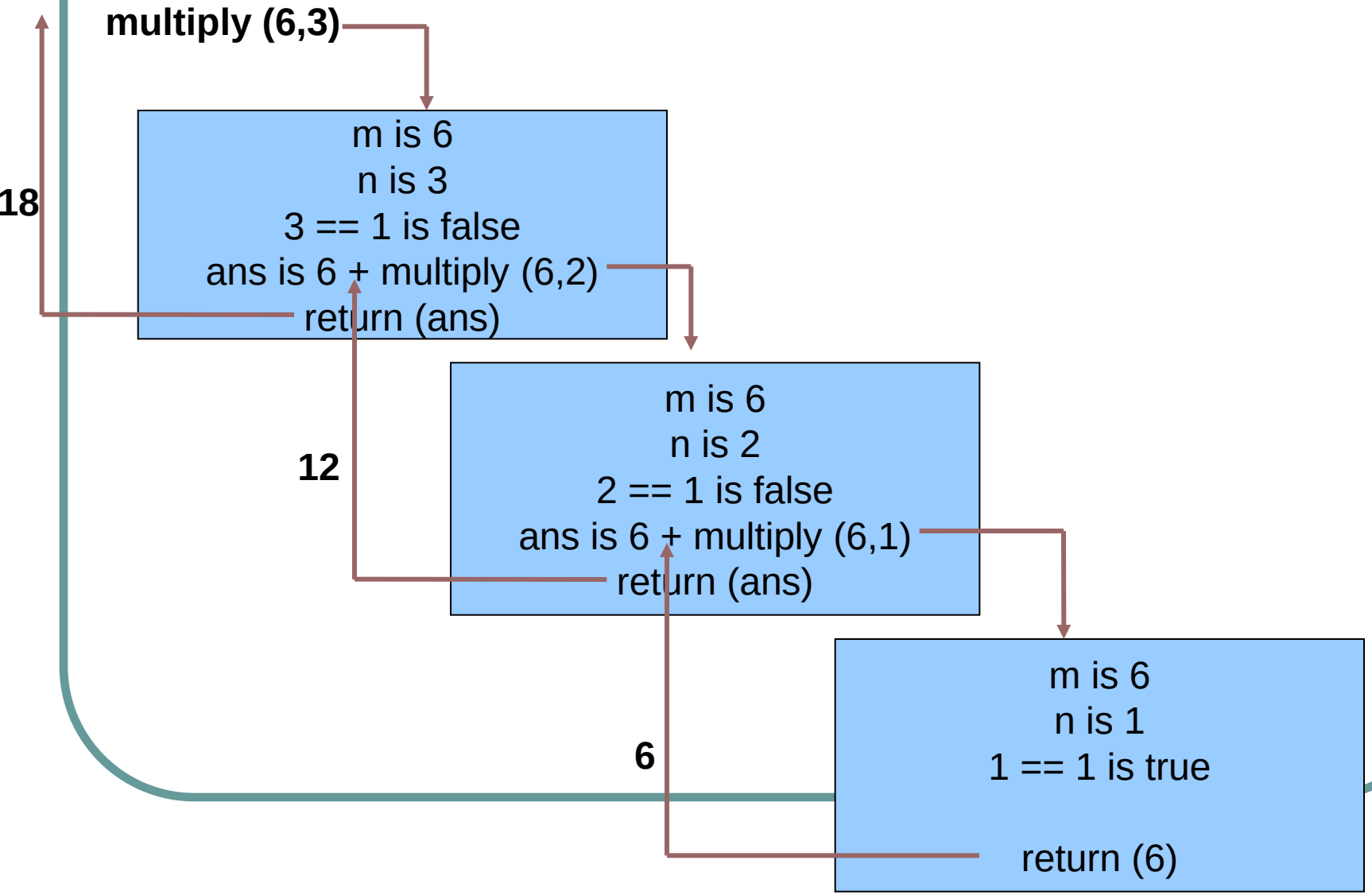
multiply(6,3) = 6 + multiply(6, 2)
= 6 + (6 + multiply(6, 1))
= 6+ (6 + (6)

Expansion phase

= 6 + (6 + (6))
= 6 + (12)
= 18

Substitution phase

Recursive Functions



Factorial

factorial(4) = 4 * factorial (3)
= 4 * (3 * factorial (2))
= 4 * (3 * (2 * factorial (1)))
= 4 * (3 * (2 * (1 * factorial (0))))

Expansion phase

= 4 * (3 * (2 * (1 * 1)))
= 4 * (3 * (2 * 1))
= 4 * (3 * 2)
= 4 * 6
= 24

Substitution phase

Recursive Functions

- ***Factorial (Recursive)***

```
Int Factr (int n)
{
int answer;
if (n==1) return(1);
answer=factr(n-1)*n;
return (answer)
}
```

- ***Factorial (non-recursive)***

```
Int Fact (int n)
{ int t, answer;
answer = 1;
for (t=1; t<n; t++)
answer= answer * (t);
return (answer);
}
```

Example 3: Power function

- Suppose we wish to define our own power function that raise a double number to the power of a non-negative integer exponent. x^n , n>=0.
- The base case is if n is 0. The answer is 1.
- The recursive case is: $x^n = x * x^{n-1}$.

$$x^n \begin{cases} 1, & n = 0 \\ x * x^{n-1}, & n>0 \end{cases}$$

Example 3: Power function …

```
double pow(double x, int n)
{
      if (n == 0)
          return 1;     /* simple case */
     else
          return x * pow(x, n  - 1);  /* recursive step */
}
```