Writing Recurrences as Functions

- Here are the 4 steps to designing a recursive solution.
  1. Define the problem in terms of a smaller version of itself.
  2. Make sure that the problem size decreases with each recursive call.
  3. Identify the base case
  4. Make sure that, given step 2, the base case will eventually be reached.

A Classic Example

- Perhaps the most widely used classical example of a recursive algorithm/problem/solution is the factorial problem.
- Factorial, as you know is described by the general equation

$$\boxed{n! = n*(n-1)*(n-2)*(n-3)*\ldots*3*2*1}$$

- For example **5! = 5 * 4 * 3 * 2 * 1**
  - Or 120 for those of you keeping score
- Now this is pretty simple in concept, and fairly straight forward to code iteratively (using a loop)

A Classic Example(2)

```
n = some integer;          // number to be factorialized
while(n > 1)
{  // begin while
     fact += n * (n-1);
     n--;
}  // end while
```

- If you follow the code here you will see that this loop does indeed calculate the factorial of any number entered into the variable n.
- Now let’s look at its recursive cousin.

A Classic Example(3)

- First the logic of the recursive solution.

1. Define the problem in terms of smaller versions of itself
    - If **5! = 5 * 4 * 3 * 2 * 1** then it can also be said that
    - **5! = 5 * 4!** And
    - **4! = 4 * 3!**
    - **3! = 3 * 2!**
    - **2! = 2 * 1!**
    - **1! = 1 * 0!**
    - And **0! = 1** (this is the base case)
    - So it is obvious that 5! Can be easily solved if we know 4! And so on .

A Classic Example(4)

2. How does each recursive call diminish in size?
   - Each recursive call reduces n by 1.
   - 5! = 5 * 4!
   - 4! = 4 * 3! And so on.
3. What instance of the problem will serve as the base case?
   - 0! = 1 is about as trivial as they come so this can easily be used as the base case.
   - Could also use 1! In this case.
4. Will the base case be reached eventually?
   - It is obvious by inspection that starting at n and decrementing n with each successive call to the function n will eventually reach 0

A Classic Example(5)

- Now let's see what the code looks like for this simple yet elegant example of a recursive solution.

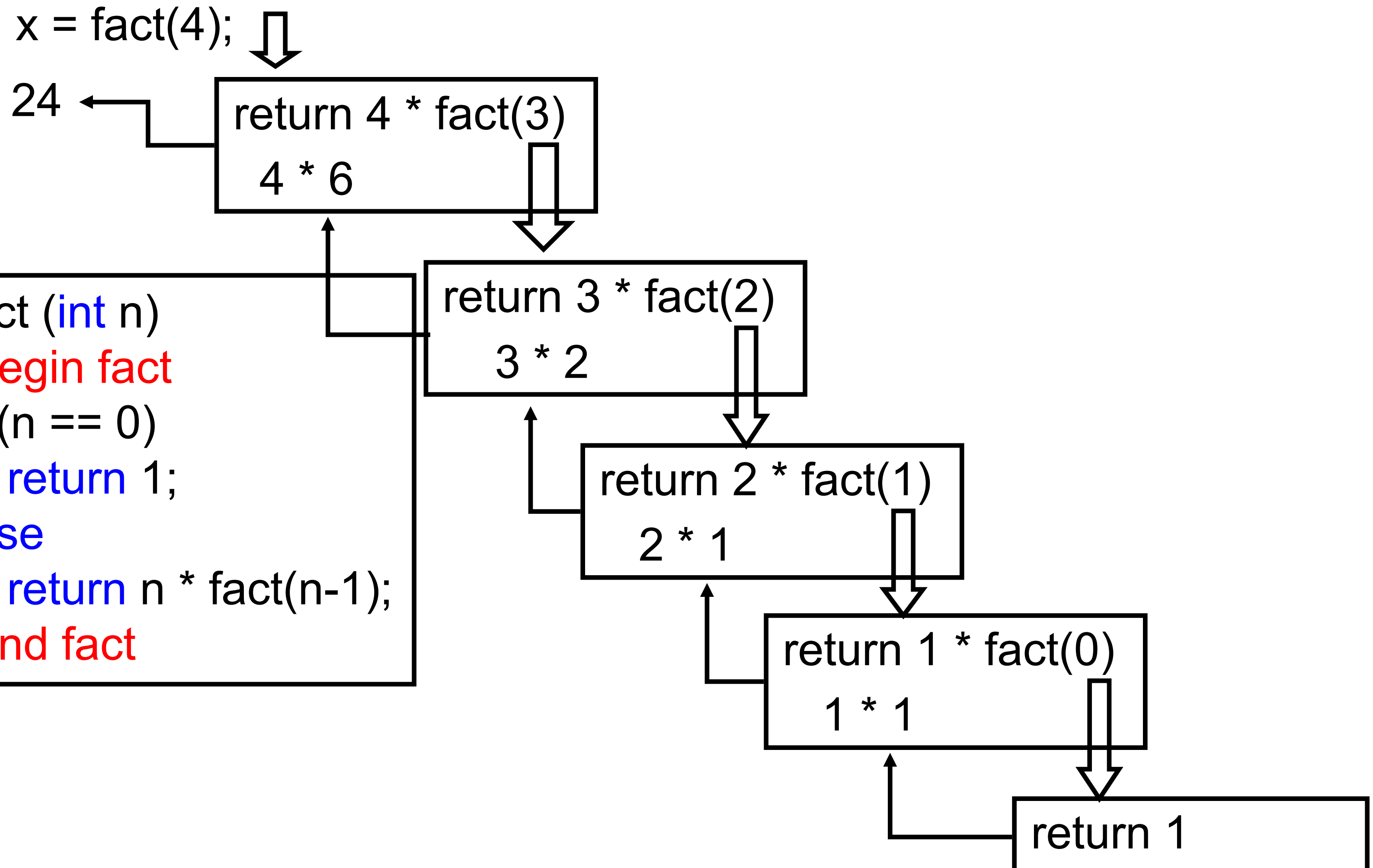
```
int factorial (int n)
{  // begin fact
    if (n == 0)
        return 1;
    else
        return n * factorial(n-1);
}  // end fact
```

Base Case

Recurrence Relation or Recursive case

A Classic Example(6)

- Here’s how the call to this function will work for the statement




```
int fact (int n)
{  // begin fact
     if (n == 0)
        return 1;
     else
        return n * fact(n-1);
}  // end fact
```