Recursion

- What happens when a function calls itself?
  - This is known as a *recursive function*

Recursion

- What happens when a function calls itself?
  - This is known as a *recursive function*
- Every function call is added to the *stack*
  - Memory is allocated
  - Control is passed to the function
- When a function ends, it is cleared off the stack
  - Memory is released
  - Control is passed back to the calling function

Recursion

- What happens when a function calls itself?
  - This is known as a *recursive function*
- Every function call is added to the *stack*
  - Memory is allocated
  - Control is passed to the function
- When a function ends, it is cleared off the stack
  - Memory is released
  - Control is passed back to the calling function
- In the simplest case, a self-calling function keeps adding to the stack (and never clearing)
  - This is *infinite recursion* (it never stops)
  - In practice, it stops when it runs out of memory and crashes

Make it Stop!

- How do you make a recursive function stop?
  - Let's say have it call itself 10 times

Make it Stop!

- How do you make a recursive function stop?
  - Let's say have it call itself 10 times
- The self call **must** be conditional
  - If can't happen every time
  - Has to happen until some condition is false
- What condition? For this one:
  - Could use a global counter
    - Not much point doing this, which we'll see in a minute
  - Could increment a counter and pass it to itself

Why Would You Do Such a Thing?

- Recursion is a useful problem-solving pattern
  - Any iteration can be done as recursion
  - Any recursion can be done as iteration
  - Some problems fit one or the other better

Example: Factorial

- The factorial of a whole number is the product of that number and all positive whole numbers less than it

```
5! = 5 * 4 * 3 * 2 * 1
```

- This can be defined recursively by two rules:

```
1) 0! = 1
2) n! = n * (n-1)! where n > 0
```

- (1) is the *base case*
- (2) is the *general case*

Example: Factorial

- A recursive factorial function:

```
int rec_fact( int n )
{
   if( n == 0 )
        // base case
        return 1;
   else
        // general case
        return n * rec_fact( n - 1 );
}
```

Example: Factorial

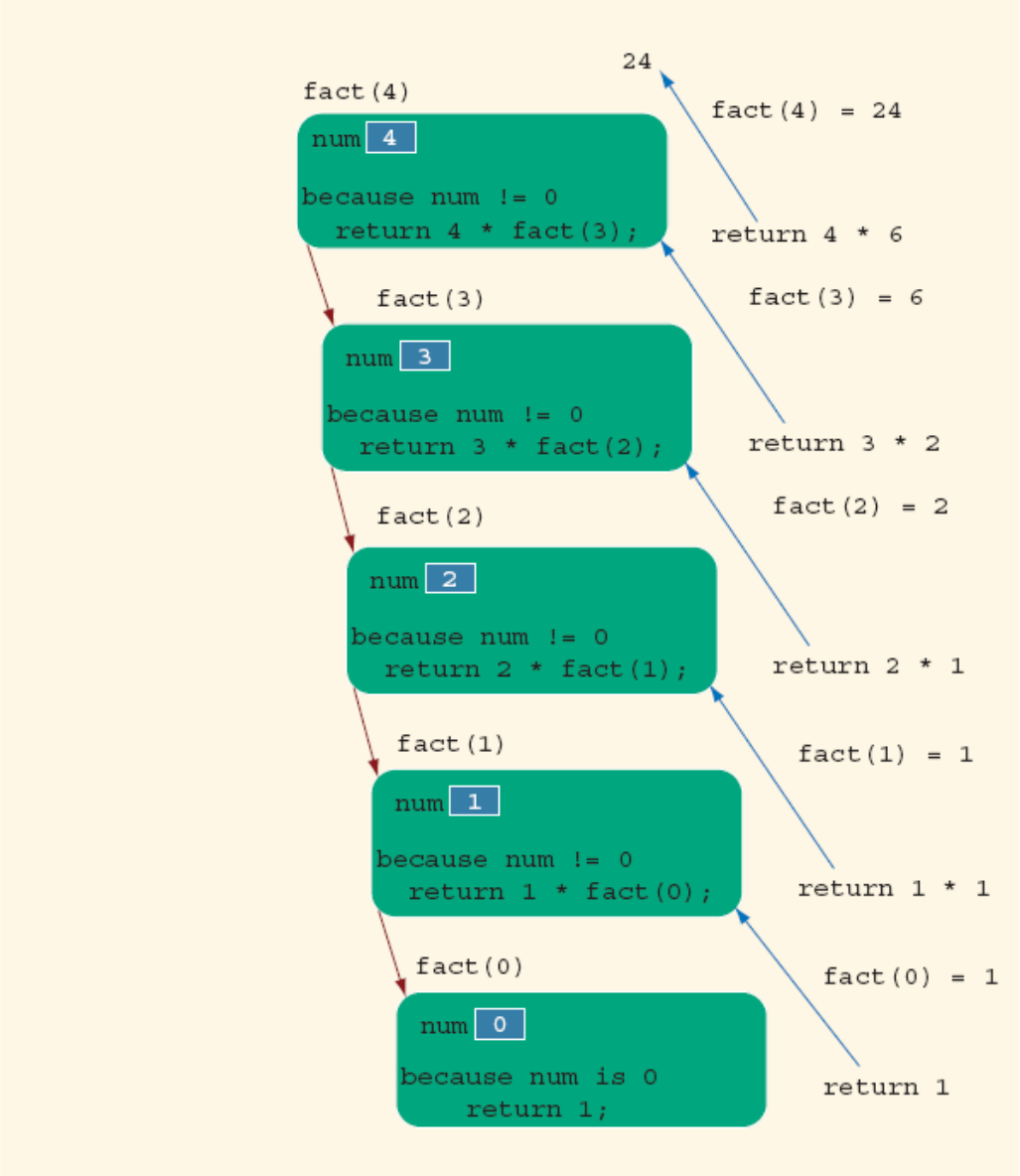

FIGURE 16-1 Execution of `fact(4)`

Exercise: Largest Number

- Recursive problems can be broken into smaller versions of themselves
  - Find the largest of these 1000 numbers
    - Find the largest of the first 500
    - Find the largest of the second 500
    - Find the largest of those 2
- Better yet, take them one at a time:
  - Find the largest of these 1000 numbers
    - Is the first the largest? Compare it with:
      - Find the largest of these 999 other numbers
        - » Is the first the largest? Compare it with:
          - Find the largest of these 998 other numbers
            - etc.

Exercise: Largest Number

- Write a recursive function to return the largest number in an array of ints
  - What is the base case?
  - What is the general case?

Exercise: Largest Number

- Write a recursive function to return the largest number in an array of ints

```
int largest( const int a[], int start, int end )
{
    if(         )
        // base case

    else
        // general case (should make a recursive call)

}
```

Exercise: Largest Number

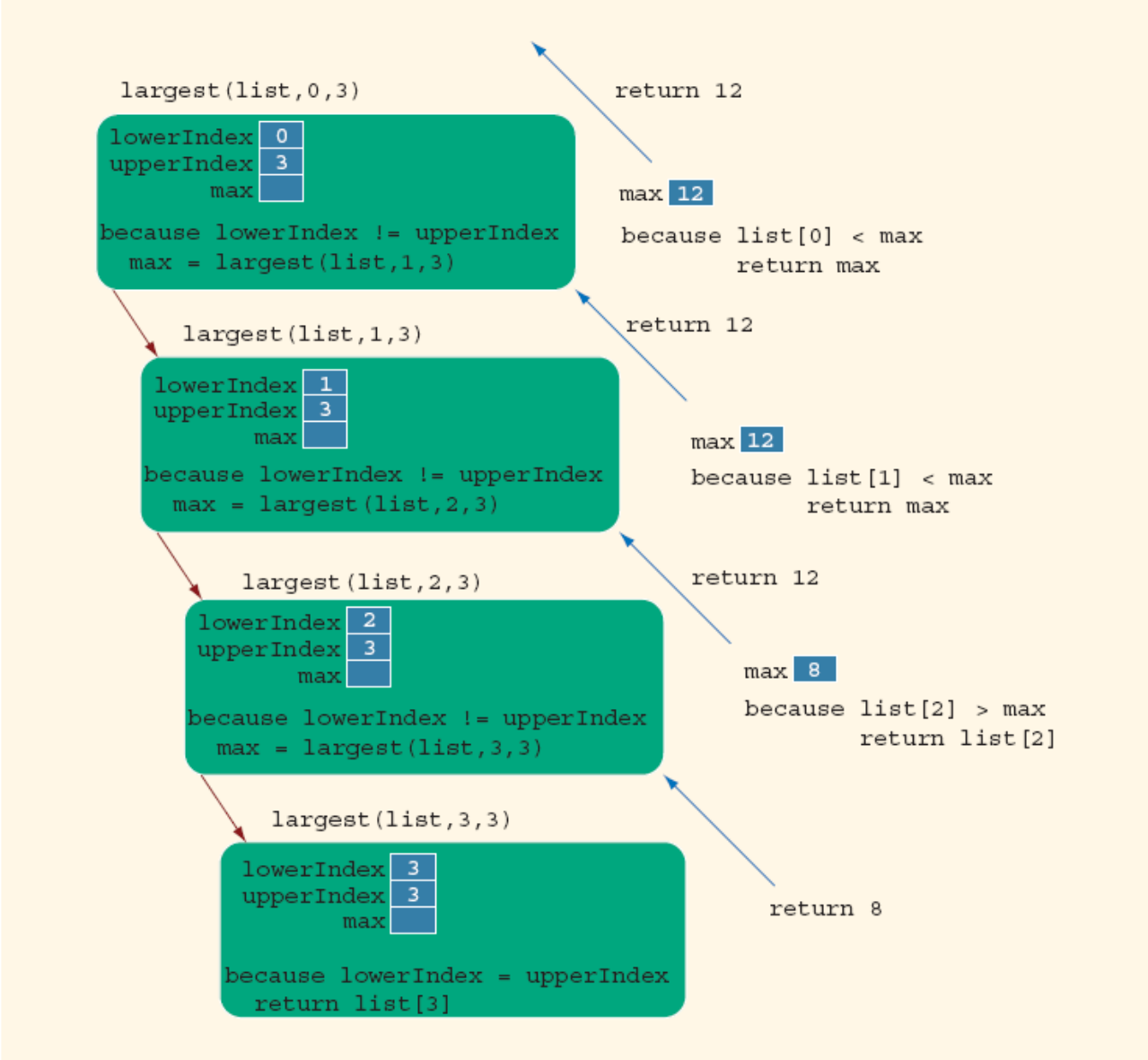

FIGURE 16-4 Execution of `largest(list, 0, 3)`

Exercise: Print a Number

- Given a positive integer, print it to the screen with spaces between each digit
  - 10578 should print “1 0 5 7 8”
- What is the algorithm?
  - (Not code, steps we will take to solve the problem)

Exercise: Print a Number

- Given a positive integer, print it to the screen with spaces between each digit
  - 10578 should print “1 0 5 7 8”
- Each digit in the integer needs to be printed
  - Do it recursively
    - What is the base case?

Exercise: Print a Number

- Given a positive integer, print it to the screen with spaces between each digit
  - 10578 should print “1 0 5 7 8”
- Each digit in the integer needs to be printed
  - Do it recursively
    - What is the base case?
      - A single digit (integer is < 10)
      - Just print that number
    - What is the general case?

Exercise: Print a Number

- Given a positive integer, print it to the screen with spaces between each digit
  - 10578 should print “1 0 5 7 8”
- Each digit in the integer needs to be printed
  - Do it recursively
    - What is the base case?
      - A single digit (integer is < 10)
      - Just print that number
    - What is the general case?
      - More that one digit (integer > 9)
      - Print the first digit
      - Print a space
      - Print the rest of the digits

Exercise: Print a Number

- Given a positive integer, print it to the screen with spaces between each digit
  - 10578 should print “1 0 5 7 8”
- Each digit in the integer needs to be printed
  - Do it recursively
    - What is the base case?
      - A single digit (integer is < 10)
      - Just print that number
    - What is the general case?
      - More that one digit (integer > 9)
      - Print the first digit
      - Print a space
      - Print the rest of the digits
  - What are the arguments, return value?

Printing Numbers

- This process we just did is what the insertion operator does for you every time you print a number
  - Breaks the number into digits
  - Prints them out as characters, one at a time
- ASCII codes:

ASCII Code	Character	ASCII Code	Character
48	'0'	53	'5'
49	'1'	54	'6'
50	'2'	55	'7'
51	'3'	56	'8'
52	'4'	57	'9'