What is Recursion?

- A recursive method is one that calls itself

Application of recursion

Recursion characteristics

- One or more base cases are used to stop recursion
- Each recursive call reduces the original problem to smaller same sub problem

- Each recursive call progresses towards a base case until it becomes that case

Example 1 - Factorials

$0! = 1$ (By Definition!)

$n! = n \times (n-1)!$ If $n > 0$

$3! = 3 \times 2!$

$2! = 2 \times 1!$

$1! = 1 \times 0!$

$0! = 1$ (Base Case!)

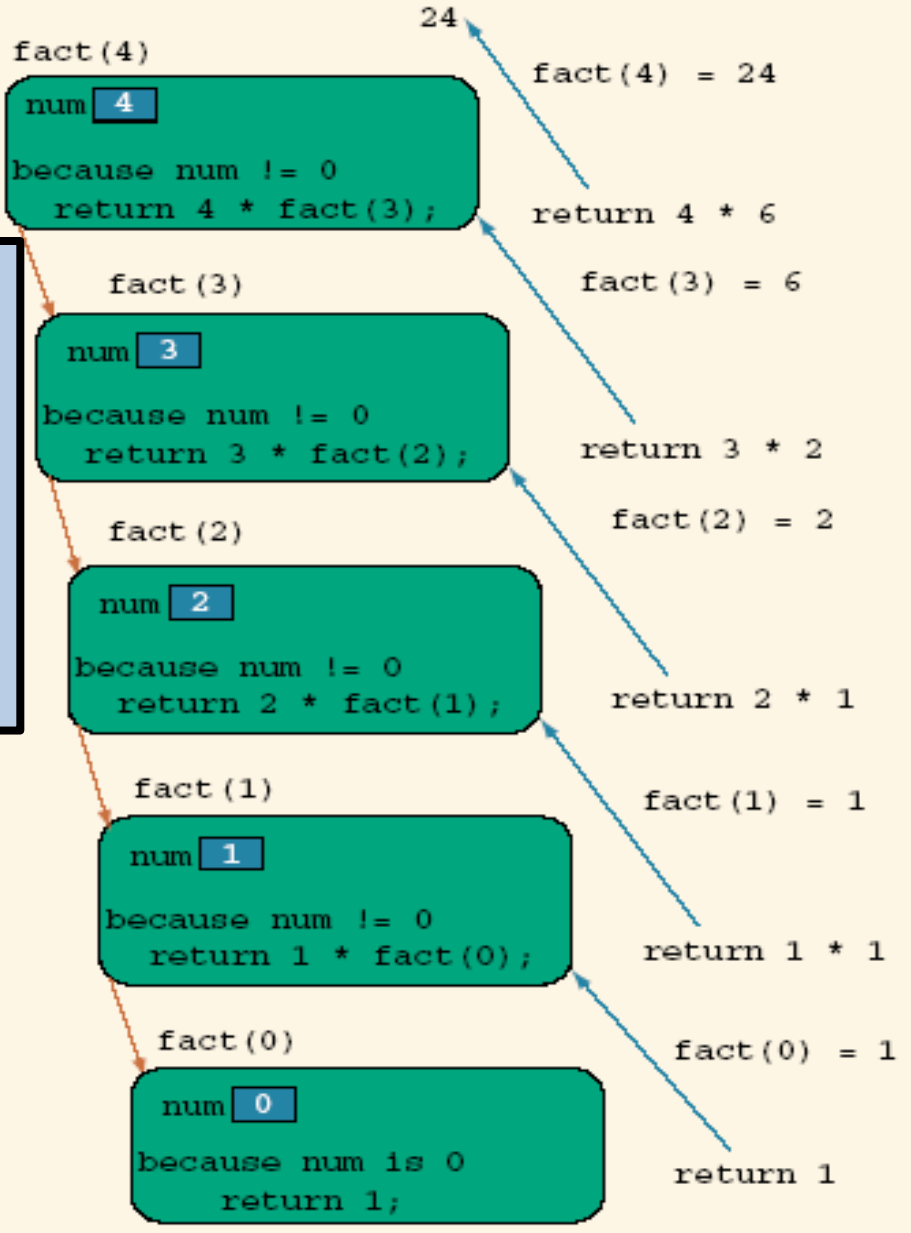
$1! = 1 \times 0! = 1 \times 1 = 1$

$2! = 2 \times 1! = 2 \times 1 = 2$

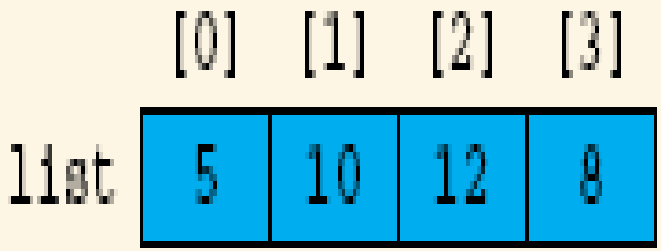
$3! = 3 \times 2! = 3 \times 2 = 6$

Example 1 - Factorials

```
public static int fact(int num) {
    if (num == 0)
        return 1;
    else
        return num * fact(num – 1);
```



Example 2 – Find maximum

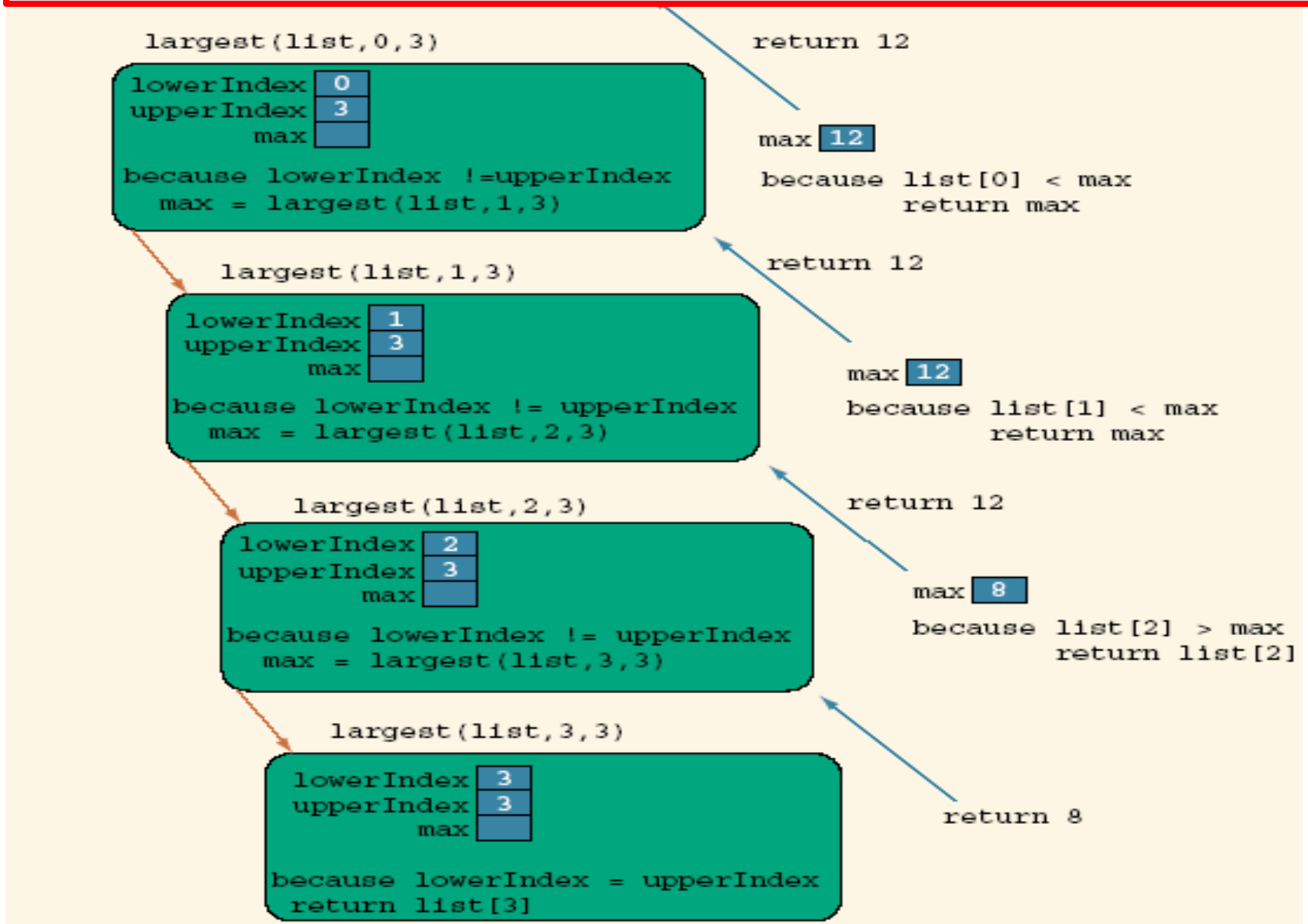


- **Integers stored in array named *list* in arbitrary order**

Example 2 – Find maximum

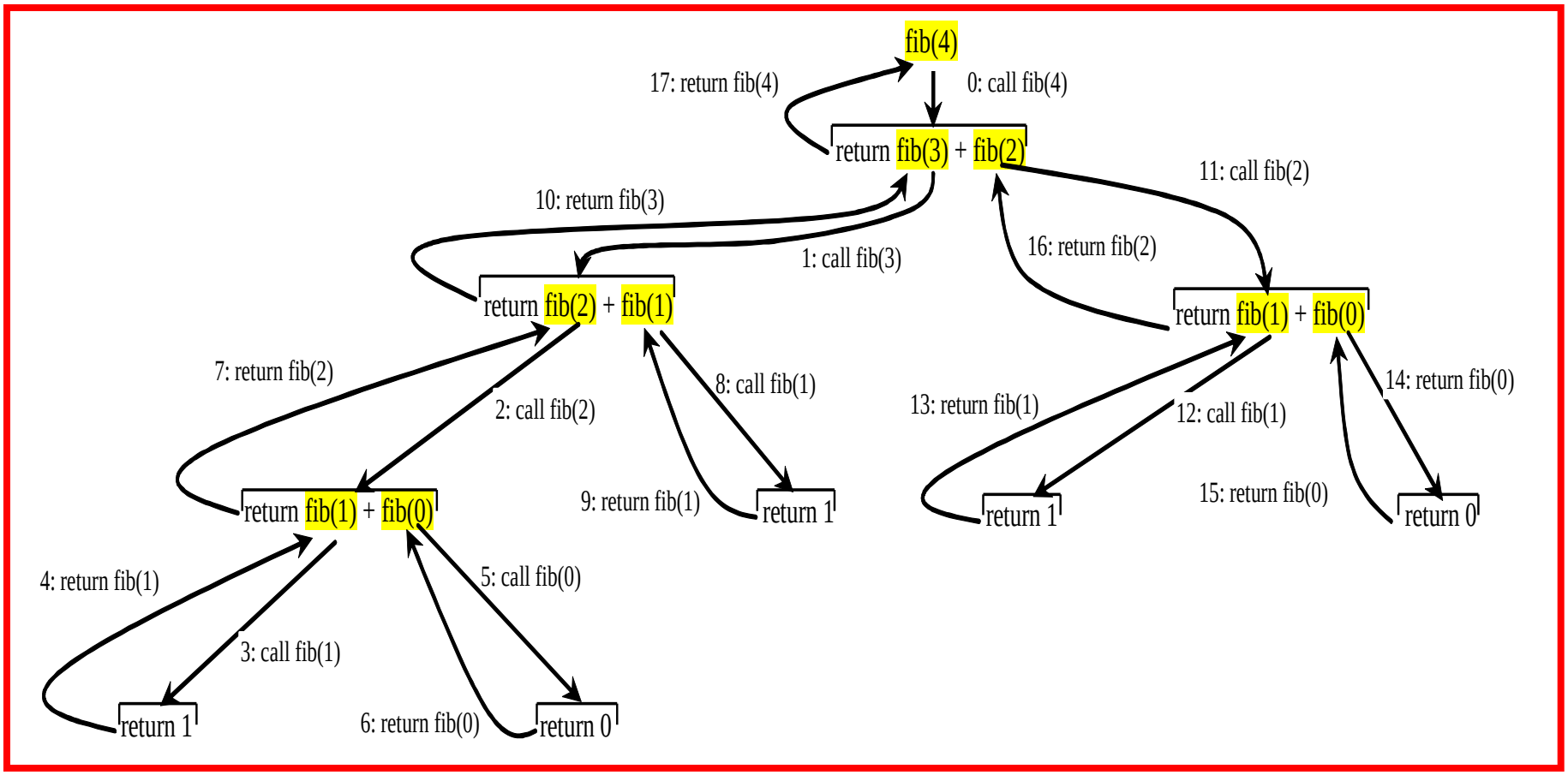
```
public static int largest(int[] list, int lowerIndex, int upperIndex) {
  int max;
  if (lowerIndex == upperIndex)
    return list[lowerIndex];
  else   {
    max = largest(list, lowerIndex + 1,
                  upperIndex);
    if (list[lowerIndex] >= max)
       return list[lowerIndex];
    else
       return max;
  }
}
```

Example 2 – Find maximum



```
public class FibonacciPerf {
    public static long FiboIterative(int n) {
        int f0 = 0, f1 = 1,
        int currentFib;
        if (n == 0) return 0;
        if (n == 1) return 1;
       for (int i = 2; i <= n; i++) {
            currentFib = f0 + f1;
            f0 = f1;
            f1 = currentFib;
        }
        return f1;
     }
```

```
public static int FiboRecursive(int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return FiboRecursive(n - 1) + FiboRecursive(n - 2);
}
```


fib(4)
17: return fib(4)
0: call fib(4)
return fib(3) + fib(2)
11: call fib(2)
10: return fib(3)
16: return fib(2)
1: call fib(3)
return fib(2) + fib(1)
return fib(1) + fib(0)
7: return fib(2)
8: call fib(1)
14: return fib(0)
13: return fib(1)
2: call fib(2)
12: call fib(1)
return fib(1) + fib(0)
9: return fib(1)
return 1
15: return fib(0)
return 1
return 0
4: return fib(1)
5: call fib(0)
3: call fib(1)
return 1
6: return fib(0)
return 0