6.00 Introduction to Computer Science and Programming
Fall 2008

6.00 Handout, Lecture 13
(Not intended to make sense outside of lecture)

```
def fib(n):
    global numCalls
    numCalls += 1
    print 'fib called with', n
    if n <= 1: return n
    else: return fib(n-1) + fib(n-2)
```

```
def fib1(n):
    memo = {0:0, 1:1}
    return fastFib(n, memo)
```

```
def fastFib(n, memo):
    global numCalls
    numCalls += 1
    print 'fib1 called with', n
    if not n in memo:
        memo[n] = fastFib(n-1, memo) + fastFib(n-2, memo)
    return memo[n]

def maxVal(w, v, i, aW):
    #print 'maxVal called with:', i, aW
    global numCalls
    numCalls += 1
    if i == 0:
        if w[i] <= aW: return v[i]
        else: return 0
    without_i = maxVal(w, v, i-1, aW)
    if w[i] > aW: return without_i
    else: with_i = v[i] + maxVal(w, v, i-1, aW - w[i])
    return max(with_i, without_i)
```

```
def fastMaxVal(w, v, i, aW, m):
    global numCalls
    numCalls += 1
    try: return m[(i, aW)]
    except KeyError:
        if i == 0:
            if w[i] <= aW:
                m[(i, aW)] = v[i]
                return v[i]
            else:
                m[(i, aW)] = 0
                return 0
        without_i = fastMaxVal(w, v, i-1, aW, m)
        if w[i] > aW:
            m[(i, aW)] = without_i
            return without_i
        else: with_i = v[i] + fastMaxVal(w, v, i-1, aW - w[i], m)
        res = max(with_i, without_i)
        m[(i, aW)] = res
        return res
```

```
def maxVal0(w, v, i, aW):
    m = {}
    return fastMaxVal(w, v, i, aW, m)
```