These problems are from a quiz given in 2012. You may check your answers by doing an experiment similar to what you did in Assignment 4. You may also compare opinions on the Moodle "Practice Problem Solutions" forum.

For each of the following functions, choose the answer that best describes the function's worst-case complexity.

Reminders:
- Combining two lists or strings with "+" takes time proportional to the combined length of the operands.
- Taking a slice of a list or string takes time proportional to the length of the slice.

A.  
```python
def guess(word):
    if len(word) < 2:
        return word
    else:
        if 'a' <= word[0] <= 'm':
            return word[0] + guess(word[1:])
        else:
            return guess(word[0]+word[2:]) + word[1]
```

N is the length of word.

O(1)  O(log N)  O(N)  O(N log N)  O(N^2)  O(N^3)  O(2^N)

B.  
```python
def december(N):
    if N < 1:
        return 0
    else:
        return N + 3*december(N-2)
```

O(1)  O(log N)  O(N)  O(N log N)  O(N^2)  O(N^3)  O(2^N)

(continued on next page)
C.
def endOfTerm(nums):
    N = len(nums) # problem size
    if N == 0:
        return 0
    elif N == 1:
        return nums[0]
    elif nums[0] < 10:
        return endOfTerm(nums[:N//2])
    else:
        return endOfTerm(nums[N//2:])

O(1)    O(log N)    O(N)    O(N log N)    O(N^2)    O(N^3)    O(2^N)

D.
def travel(nums):
    N = len(nums) # problem size
    if N == 0:
        print "X"
    if N == 1:
        print nums[0]
    else:
        travel(nums[1:])
        print "Y"
        travel(nums[:-1])

O(1)    O(log N)    O(N)    O(N log N)    O(N^2)    O(N^3)    O(2^N)

E.
def snowman(nums):
    N = len(nums) # problem size
    if N == 0:
        return 0
    elif N == 1:
        return nums[0]
    else:
        return snowman(nums[:N//2]) + snowman(nums[N//2:])

O(1)    O(log N)    O(N)    O(N log N)    O(N^2)    O(N^3)    O(2^N)