Activities: More on Dictionaries

1. The Lucas numbers are defined by the following formula.

$$L_n = \begin{cases} 2 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ L_{n-1} + L_{n-2} & \text{if } n > 1. \end{cases}$$

   The Lucas number sequence is 2, 1, 3, 4, 7, 11, 18, 29, … (and has number A000032 in the Online Encyclopedia of Integer Sequences). Implement the Lucas number in the "inane" way and with memoization.

2. The Josephus numbers are defined by the following formula:

$$J_n = \begin{cases} 2 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ 1 & \text{if } n = 2; \\ 2 \times J_m - 1 & \text{if } n > 1 \text{ and } n = 2m\,. \\ 2 \times J_m + 1 & \text{if } n > 1 \text{ and } n = 2m + 1. \end{cases}$$

   Thus, $J_3 = 2 * J_1 + 1 = 3$ since 3 is odd and $J_4 = 2 \times J_2 - 1 = 2 \times 1 - 1 = 1$. Use memoization to implement this function efficiently.

3. The binomial of two integers $\binom{n}{k}$ is the number of times that k elements can be chosen out of n different elements. It gets its name because the binomials appear as the coefficients of the expansion of $(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}$. For quick calculation of the binomial coefficients, we use the formula $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$. The base cases are given by $\binom{n}{0} = \binom{n}{n} = 1$, and if $k < 0$ or $k > n$ then we have $\binom{n}{k} = 0$.

In order to use memoization, we need to store the interim results in a dictionary. However, a dictionary needs keys that are immutable. Luckily, tuples come to our rescue. A tuple is a an *immutable* list of items. It is defined by using round parentheses around the list, as in (5,2). We form the two parameters of the binomial function into a 2-tuple and use it as the key. If we have calculated `binomial(l,k)`, we store it in the dictionary `dbin` as

```
dbin[(l,k)] = ret_value
```

Notice the parentheses around the parentheses inside the bracket.

Implement a memoized, recursive version of the binomial function of the two parameters *n* and *k*.

PS: If you try this for larger numbers, you might get a "recursion depth exceeded" message. This is because each call to a function necessitates putting information on the stack, the so-called frame. The stack is stored in memory and too many pending function calls can lead to using up all memory and to sudden, very slow performance of the system. If you want to push your systems closer to its limits, you can `import sys` and use `sys.setrecursionlimit(10000)`.

4. Use a counter to find the most common word in the mobydick.txt file. Make sure you strip and lower all your words.