10/11/07 Name: _____

Divide-and-Conquer Idea: Solve problem by:

- dividing it into small problem(s) (maybe smaller instances of the same problem)
- solving the smaller problem(s)
- use solution(s) to smaller problem(s) to solve original problem

1) The recursive definition of the Fibonacci sequence (0, 1, 1, 2, 3, 5, 8, 13, ...) is:

$$f_0 = 0$$

$$f_1 = 1$$

$$f_n = f_{n-1} + f_{n-2} \quad \text{for } n \geq 2$$

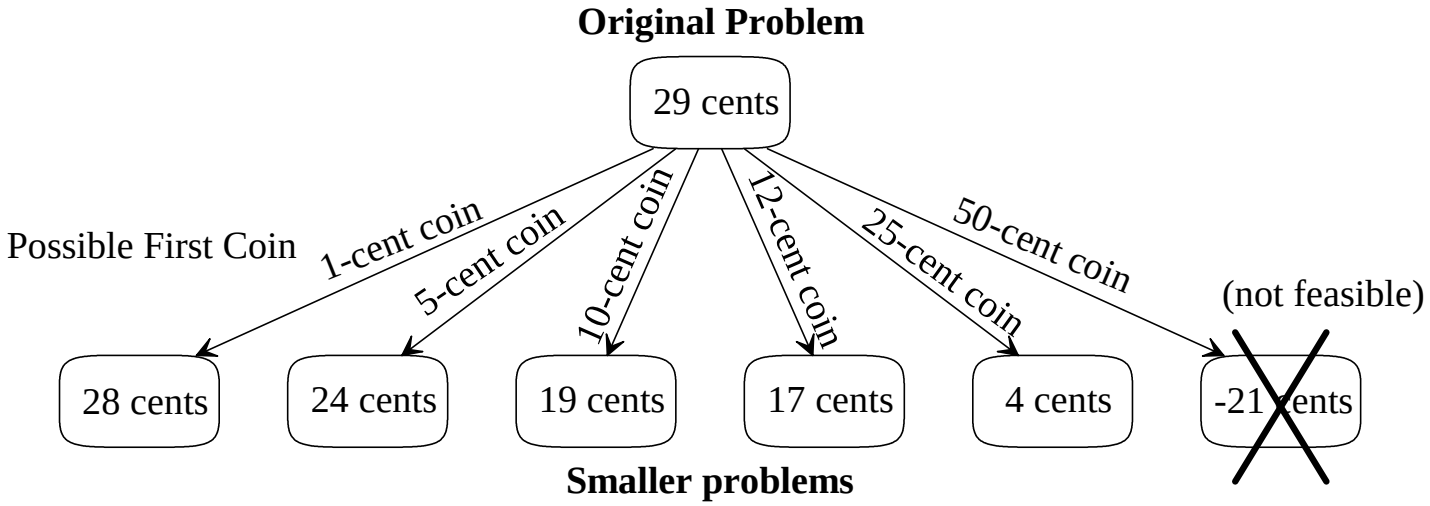
The straightforward, recursive, divide-and-conquer algorithm that calculates the n^{th} element in the sequence is:

```
int fib( int n ) {
     if ( n == 0) {
          return 0;
     } else if ( n == 1 ) {
          return 1;
     } else {
          return fib(n-1) + fib(n-2);
     } // end if
} // end fib
```

a) Executing fib on large `n` values takes a long time. What makes this recursive Fibonacci calculation so slow?

b) Can you think of a nonrecursive algorithm that would be faster?

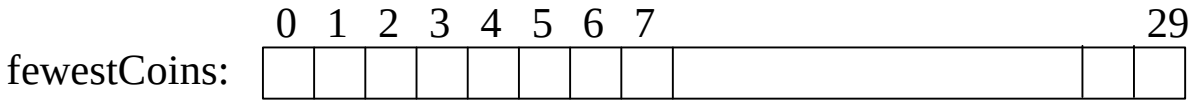
2) For the coin-change problem, after we give back the first coin, we have smaller instances of the same problem.



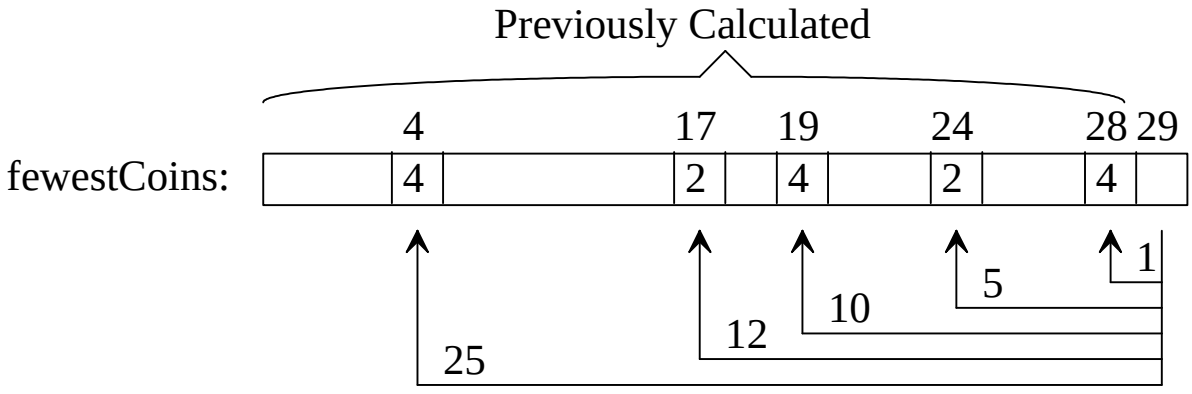
a) Why is the run-time for this simple, recursive, divide-and-conquer algorithm slow?

b). In a dynamic programming coin-change solution, you fill an array fewestCoins from 0 to the amount of change.

An element of fewestCoins stores the fewest number of coins necessary for the amount of change corresponding to its index value. For 29-cents using the set of coin types {1, 5, 10, 12, 25, 50}, determine the value of fewestCoins[0] to fewestCoins[7].



c) For 29-cents using the set of coin types {1, 5, 10, 12, 25, 50}, determine the value of fewestCoins[29].



3) Binomial Coefficient - # of combinations of "n choose k" where order does not matter (# of unique sets of size k)

$$\left[\begin{matrix} n \\ k \end{matrix} \right] = \frac{n!}{k!(n-k)!}$$

To apply dynamic programming we want to view the calculation recursively. Suppose we had a bin containing four marbles with the letters A, B, C, D, and E painted on them. The "5 choose 3" could be viewed as the sets with marble A and the sets without marble A.

a) List the sets with marble A

b) List the sets without marble A

c) Write the recursive definition for the binomial coefficient

$$\left[\begin{matrix} n \\ k \end{matrix} \right] =$$

d) To utilize dynamic programming with the binomial-coefficient problem we need to store answers to smaller size problems in an array. How many dimensions should the array be (1, 2, 3, etc.)?

e) Which part of the array will correspond to the base cases?

f) For the binomial-coefficient problem where n = 7 and k = 4, what part of the array will be needed in the calculation of the ultimate result?

g) Can the dimensionality of the array be reduced? (Justify your answer)

h) What would be the memoization algorithm for binomial coefficient look like?