1. (40 pts total) Recall that Huffman codes are constructed in a greedy fashion.

   (a) (15 pts) What is an optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

   `a:1  b:1  c:2  d:3  e:5  f:8  g:13  h:21`

   (b) (15 pt) How many optimal Huffman codes are there for this set of frequencies? Justify your answer.

   Hint: Symmetries and tie-breaking.

   (c) (10 pts) Generalize your answer to find an optimal code when the frequencies are the first n Fibonacci numbers.

   Hint: Let $F(n)$ be the nth Fibonacci number: $\sum_{i=1}^{n} F(n) = F(n+2) - 1$

2. (60 pts total) Professor Hagrid is struggling with the problem of making change for n cents using the smallest number of coins. Let the coin values be $v_1 > v_2 > \cdots > v_r$ for r coins types, and let each coin's value v_i be a positive integer. The output will be a set of counts $\{d_i\}$, one for each coin type, such that $\sum_{i=1}^{k} d_i = n$ and where k is minimized. (Note: v_1 is the *most* valuable coin.)

   (a) (20 pts) Give a greedy algorithm, that takes $O(n)$ time, to make change consisting of quarters (worth 25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent). Prove that your algorithm yields an optimal solution.

   Hint: First, consider the way you yourself solve this problem with real currency. Then, prove the greedy-choice property exists for the general problem.

   (b) (20 pts) Suppose that the available coins are in the denominations that are powers of c, i.e., denominations of $c^0, c^1, \ldots, c^\ell$ for some integers $c > 1$ and $\ell \geq 1$. Prove that the greedy algorithm always yields an optimal solution in this case.

   Hint: Consider the special case of $c = 2$, and think about what kind of counting this becomes.

   (c) (20 pts) Give a set of coin denominations for which the greedy algorithm does not yield an optimal solution, and explain why. The set should include a penny so that there is a solution for every value of n.

   Hint: The optimal solution minimizes k, the number of distinct coins used.

3. (30 pts extra credit) Let A and B be arrays of integers. Each array contains n elements, and each array is in sorted order (ascending). A and B do not share any elements in common. Give a $O(\lg n)$-time algorithm which finds the median of $A \cup B$ and prove

that it is correct. This algorithm will thus find the median of the $2n$ elements that would result from putting A and B together into one array. (Note: consider refreshing your memory of the definition in the textbook for the median of a list with an even number of elements.)