Programming Project #2
CS 3410

Use Dynamic Programming to solve the "Making Change" problem. I want you to solve the problem 3 different ways:

a. building the table bottom-up
b. recursively, without memorization (hint: it will take forever; you don't have to run for all problem sizes)
c. recursively, with memoization

Specific guidelines:

a. The specific problem to be solved will be read from a data file. This file will contain a number *n* which specifies the number of different denominations in the money system. The next *n* lines contain the numeric value (in "pennies") of the *n* denominations. Denominations will be ordered from smallest to largest. Following this is a line containing a single integer *k*, indicating how many problems must be solved, i.e., how many different money values must be solved. This is followed by *k* lines, each containing a single integer specifying a money value to be used as an input into the problem; i.e., you will have to calculate how to make change from it. These values are <u>not</u> guaranteed to be in numeric order. See sample below.
b. You can assume all problems will contain a 1 cent denomination.
c. Results can be written to file or to screen (but I need to see a copy of results).
d. You will work teams of 2.
e. You can use either C++ or java.
f. You can use any development environment you choose.
g. You must solve using dynamic programming, and the 3 methods specified.
h. I recommend you get the bottom up working first. Then get the memoization version working, then simply comment out the line which uses the memoization results to give you the non-memo version.
i. Start early … the debugging of the recursive routine can be tough.

Required for turn-in:

a. Listing of your programs (properly commented, and in compliance with the appropriate CS style guide). Include in your header an overview of your algorithm.
b. Correctness results; i.e., the output of your program for each of the 3 solutions using the sample test data.
c. Performance results from the tests you ran; as a minimum, run the test set provided below. Provide a graph of the results.
d. An analysis of the measured running time of the 3 solutions. I just need 2-3 sentences of your conclusions.
e. A CS Cover Sheet, including a signed statement regarding how well your final code worked. For example, "Our code compiled and ran properly, and produced the correct output" or "Our code compiled correctly, but core dumps when executed."
f. (optional) Comments on problems you ran into/lessons learned.

Sample Input

```
6
1
7
17
23
37
52
9
4
16
22
29
34
1033
86
188
255
```

Sample Output

```
4 cents =  1:4                  // denom=1  number =4
16 cents =  7:2  1:2        // list from biggest denomination to smallest
22 cents = 7:3  1:1
29 cents = 7:4  1:1
etc.
```