CS310 Project 3

For this project, you will solve the same problem with each of the algorithm design approaches we have learned: greedy, divide & conquer, and dynamic programming.

The main problem for this project is making change for a given amount with the fewest possible coins, using a set of possible coin values. You will modify the file **make_change.c** to complete the functions **greedy**, **divide_and_conquer**, **dyn_prog**, and **print_sol**. For each, you will need to take the following approaches:

greedy should use the highest value from **coins** that could be used to make change for **val**. It then subtracts that value away from **val** and repeats the process until **val** is zero or there is no coin amount that can be used to make change for **val**'s current value. It should return the number of coins it took to make change, or -1 if the approach fails to find a way to make change.

divide_and_conquer uses a memoized recursive approach where it loops through each coin value that can be used to make change for **val**, then calls itself recursively on the remaining amount that needs to have change made for it. It should have two base cases. When **val** is already represented in **table**, in which case it returns the value from **table**. When val is 0, then change has been found and it can return. When a recursive call has returned, its result can be compared to the other possible coin choices for **val** and store the minimum number of coins that were needed to make change in **table**.

dyn_prog must use a bottom-up approach to the problem, solving the smallest subproblems first and working its way up to **val**, storing the counts of the numbers of coins needed in **table** and referencing them to compute higher values, looking at each possible coin value that could be used for **val** and finding the one that has the minimum count stored in **table**. It should also store the coin value used to make change in **sol** so the full solution can be reconstructed later.

For example, if **dyn_prog** is called with **val** being 8 and coin amounts of 10, 5, and 1, then it should return 4 and result in the following entries being set in **table** and **sol**:

	table	sol
0	0	0
1	1	1
2	2	1
3	3	1
4	4	1
5	1	5
6	2	5
7	3	5
8	4	5

print_sol reconstructs the solution using the sol array. It looks up the entry for **val** in **sol**, printing the amount it finds in that entry, subtracting it away from **val**, and repeating until **val** is zero. It should print a line that looks like this:

```
Solution: 5 1 1 1
```

Submission:

- Submit the following file on Moodle:
  `make_change.c`
- Do not turn in executables or object code. Programs that produce compile-time errors or warnings will receive a zero mark (even if it might work perfectly on your home computer.)
- Be sure to provide comments in your program. You must include the information as the section of comments below:

```
/**    CS310 Project 3
File: XXX.c
Name: XXX
Date: XXX
Collaboration Declaration: assistance received from [TA name]
*/
```

Some notes on grading:

- Programs are graded for correctness (output results and code details), following directions and using specified features, documentation and style.
- Here is a tentative grading scheme.

Greedy	
solves correct cases	5
correct approach	10
Divide & conquer	
solves correct cases	5
correct approach	10
Dynamic programming	
solves correct cases	5
correct approach	10
Solution reconstruction	
prints correct output	5