**CS 3100 / In-class Activity 5, Making Change with Dynamic Programming**

| **Name** | **Computing ID** |
|---|---|
| *Your Name:* | |

**In class:** You must work in teams of 2, 3 or 4. Each person writes answers and turns in the sheet at end of class. **Missed class?** Work alone and answer to the best of your ability. Submit to GradeScope by 9am on the 2nd day after in-class activity.

**1.** Write the greedy choice property for the greedy algorithm for making change:

**2**. If your coin denominations are [25, 13, 7, 5, 1], find a change-amount > 5 where the greedy approach will correctly give the smallest number of coins.

**3**. If your coin denominations are [25, 13, 7, 5, 1], find a change-amount > 5 where the greedy approach will NOT correctly give the smallest number of coins.

**4**. Let's think about dynamic programming for this problem. We ask you to think about "the last thing you would do." For this problem, that would be the last coin you choose. Let's say that Change($n$) is the minimum number of coins needed to give change for $n$ cents. If you have these coin denominations [25, 11, 10, 5, 1] and a quarter is the last coin you pick, fill in the blanks to show what the value would be if we knew that quarter was part of the solution:

$$\text{Change}(n) = \text{Change}(\underline{\hspace{3cm}}) + \underline{\hspace{1.5cm}}$$

**5.** What's the value of the base case, Change($0$)?

**6**. We don't know that we'll pick a quarter or some other coin, but as we did with other dynamic programming solutions, our recursive definition of the solution will list all options and pick the best. Before we do that, note that we can't pick a quarter if $n$<25.
Below we've started to list the options for the result of choosing one of the coins. You can see the minimum of the subproblems' solutions is being chosen as the solution for the value n. We've shown that one option is choosing a quarter, but only if the amount is 25 cents or larger.
**For you to do:** list the remaining options for the other coins below the subproblem using a quarter, to the right of the bracket that's taking the min of these values. Be sure to include a "test" like with did for the quarter's case.

$$\text{Change}(n) = \min \begin{cases} \text{Change}(\underline{\hspace{3cm}}) + \underline{\hspace{1.5cm}} & \text{if } n \geq 25 \\ \\ \\ \\ \end{cases}$$

**7**. Let's start to think about a top-down algorithm that uses memoization to solve this. We'll use an array *memo* for the memoization table. If n is the amount of change and k is the number of coin denominations, what will be the size of the memoization array *memo*?

**8.** Below is incomplete Python-like code for the top-down DP solution to this. Try to fill in the blanks to make this code complete. Also, talk with your group and see if you can write the time-complexity in terms of n and k.

```
denom = [25, 10, 11, 5, 1]
amount = 15
memo = ??? # explain what this is initialized to
num_coins = change(amount, denom, memo)

def change(n, denom, memo):
   if n == 0:
      return ______

   if memo[n] ___________ :
      return ___________

   min = sys.maxsize # a really big number
   for i in range(len(denom)):
      if n >= denom[i]:
         result = change(_____________, denom, memo) + 1
         if result < min:

            _________________

   memo[n] = ______________
   return min
```