**4.5 Dynamic Programming Worksheet Homework submission**

**(all sections need to be answered)**

**Problem:**  Dynamic Programming Problem – in the “Republic of Coingo” the coin denominations are {1,2,6} for some strange historic reason. Therefore ***coinList={1,2,6}*** for this country.

The “naïve” recursive Fewest algorithm **(FC)** for generating change that has been used to find the fewest coins is shown below.

**Function FC(coinList,change)**

// assuming change can be given in 1c coins

minCoins:=change

if change in coinList then

return 1

else

foreach in coinList do

if ( <= change) then

numCoins = 1 + **FC**(coinList,())

if (numCoins < minCoins) then

minCoins := numCoins

end if

endif

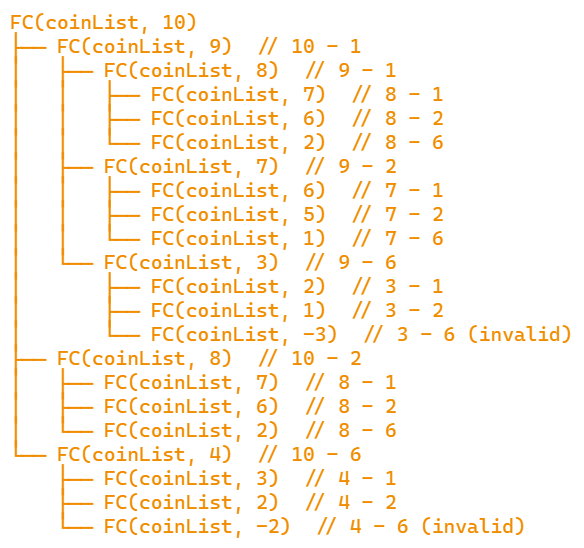
end do

end if

return minCoins

end function

1. Show a “call tree” to at least 3 levels of recursion for generating change with the fewest coins for 10c using the “naïve” Fewest coin algorithm “FC” shown above. Refer to the 4.5 Dynamic Programming Online lesson transcript and the recorded online lesson for guidance on what needs to be shown for all calls of “FC”.



1. What is the “worst case” time complexity for the “FC” algorithm? Justify your answer.

The worst-case time complexity of the FC algorithm is O(c^n), where c is the number of coin denominations and n is the amount of change to be made.

1. For each recursive call, the algorithm makes up to c recursive calls (one for each coin denomination).
2. The depth of the recursion tree can go up to n in the worst case (when using only 1-cent coins).
3. This leads to a tree with c^n nodes in the worst case.
4. Each node performs a constant amount of work
5. Describe how this problem be solved using Dynamic programming?

 Create an array dp of size change + 1 to store the minimum number of coins needed for each amount from 0 to change.

 Initialize dp[0] = 0 (it takes 0 coins to make 0 cents).

 For each amount i from 1 to change: a. Initialize dp[i] to infinity. b. For each coin c in coinList:

* If c <= i, then dp[i] = min(dp[i], 1 + dp[i-c]).

The final answer will be in dp[change

1. Create an array from 1 to 10c of 10 cells and using Dynamic programming methods fill in the fewest coins from 1 to 10 cents inclusive.

Amount: 0 1 2 3 4 5 6 7 8 9 10

Coins: 0 1 1 2 2 3 1 2 2 3 2