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

The worst case time complexity of the FC algorithm is exponential, because at each layer, FC is called again for every coin in coinList. Therefore, the Big O time is where L is the length of the coinList and C is the amount of change to be given.

1. Describe how this problem be solved using Dynamic programming?

This problem can be solved with dynamic programming because of the many overlapping problems in the call tree. For example, FC({1, 2, 4}, 9) requires FC({1, 2, 4}, 8) as well, so it would be wasteful to recompute it again. As such, we can store the values of previous FC calls so that they can subsequently be accessed in O(1) time.

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.

I’m a bit unsure what the question is asking me to do