A. (Forward) Print

Base Case: If the node reference points to *null*, we are at the *end of the list*, so *terminate* the method

Recursive Case:
1. Print the *current node’s data*
2. Print the *remainder* of the list *(next node, onwards)* via a recursive call

B. Reverse Print

Base Case: If the node reference points to *null*, we are at the *end of the list*, so *terminate* the method

Recursive Case:
1. Print the *remainder* of the list *(next node, onwards)* via a recursive call
2. Print the *current node’s data*

C. Sum of all Money objects

Base Case: If the node reference points to *null*, we are at the *end of the list*, so return 0 (no Money value)

Recursive Case:
1. Get the *current node’s Money value*
2. Return the current node’s money value plus the sum of the *remainder* of the list (recursive call)

D. Sum of all Quarter objects

Base Case: If the node reference points to *null*, we are at the *end of the list*, so return 0 (no Money value)

Recursive Case:
1. Check if *current node has a Quarter object*
2. Return .25 (or 0 if current node doesn’t contain Quarter object) plus sum of the *remaining* Quarters in the list (recursive call)

Note(s)
- Do not make the linked list’s nodes public just to make the recursive methods work from main()
- Suggested approach:
  ```java
  private static void print(MoneyNode node) {
      //“static” can be omitted
      ...
  } //helper method, for internal-use only, all the recursion happens here
  
  public void print() {
      print(first.next); //call private helper method
  } //public method that the user can call from main()
  ```