Answer for SET5

(1)

Algorithm: L2BST(L)

Requires: a sorted list of numbers

Return: a minimal balanced binary search tree

```
1. if isEmpty(L) then
2.     return leaf
3. elseif isEmpty(tail(L)) then
4.     return node(leaf,head(L),leaf)
5. Else
6.     Let n = length(L)
7.     Let list1 = cut(L,n/2,n)
8.     Let list2 = cut(L,1,n/2)
9.     return node(L2BST(list1),getNth(n/2,L),L2BST(list2))
10.endif
```

trace for L=[2,4,6,8,10]

```
L2BST([2,4,6,8,10])
   isEmpty([4,6,8,10])??                    NO!
   return node(L2BST([2,4]),6,L2BST([8,10]))
L2BST([2,4])
   isEmpty([4])??                           NO!
   return node(L2BST([]),2,L2BST([4]))
L2BST([4])
   isEmpty([])??                            YES!
   return node(leaf,4,leaf)
L2BST([8,10])
   isEmpty([10])??                          NO!
   return node(L2BST([]),8,L2BST([10]))
```

```
L2BST([10])
   isEmpty([])??                                YES!
   return node(leaf,10,leaf)
```

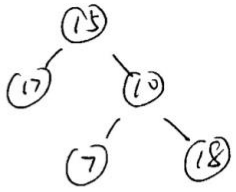
(2)

Algorithm: delRoot(binT)

Requires: a BT

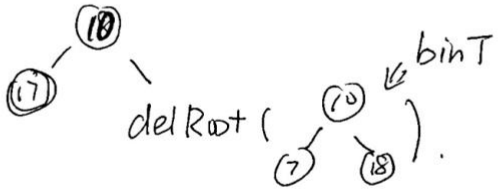
Return: a new BT whose root is the root of the right subtree

```
1. if isLeaf(binT) then
2.     return leaf
3. elseif isLeaf(left(binT)) && isLeaf(right(binT)) then
4.     return leaf
5. elseif isLeaf(left(binT)) then
6.     return right(binT)
7. elseif isLeaf(right(binT)) then
8.     return left(binT)
9. else
10.    return
   node(left(binT),root(right(binT)),delRoot(right(binT)))
11.endif
```

2. Trace for  ⇐ binT.

delRoot (binT)

isLeaf ?? isLeaf(left) ?? isLeaf(right) ?? NO!

return. 

delRoot (right (binT))

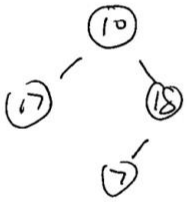
isLeaf ? isLeaf (left) ?? isLeaf (right) ?? NO!

return. 

delRoot (18)

isLeaf (left) && isLeaf (right) ?? YES!

return leaf.

=> delRoot (binT) =>  ✓

(3)

Algorithm: belX(x,BST)

Requires: a BST and a value x

Return: counts the number of nodes whose values are less than x

```
1. if isLeaf(BST) then
2.     return 0
3. elseif x == root(BST) then
4.     return treeSize(left(BST))
5. elseif x < root(BST) then
6.     return belX(x,left(BST))
7. else
8.     return 1 + treeSize(left(BST))+ belX(x,right(BST))
9. endif
```

trace for belX(20,T)

```
belX(20,T)                                              4
    isLeaf(T)??                         NO!
    20 < 50
    return belX(20,left(T))
belX(20,left(T))                                        4
    isLeaf(left(T))??                   NO!
    20 > 17
    return 1 + treeSize(left(T))+ belX(20,right(left(T)))
belX(20,right(left(T)))                                 0
    isLeaf(right(left(T)))??            YES!
    return 0
```

trace for belX(58,T)

```
belX(58,T)
    isLeaf(T)??                              NO!
    58 > 50
    return 1 + treeSize(left(T))+ belX(58,right(T))
belX(58,right(T)
    isLeaf(right(T))??                       NO!
    58 < 60
    return belX(left(right(T)))
belX(left(right(T)))
    isLeaf(left(right(T)))??                 NO!
    58 > 55
    return       1       +       treeSize(left(left(right(T))))+
belX(58,right(left(right(T))))
belX(58,right(left(right(T))))
    isLeaf(right(left(right(T))))??          NO!
    58 > 57
    return     1     +     treeSize(left(right(left(right(T)))))+
belX(58,right(right(left(right(T)))))
belX(58,right(right(left(right(T)))))
    isLeaf(right(right(left(right(T)))))??   YES!
    return 0
```

(4)

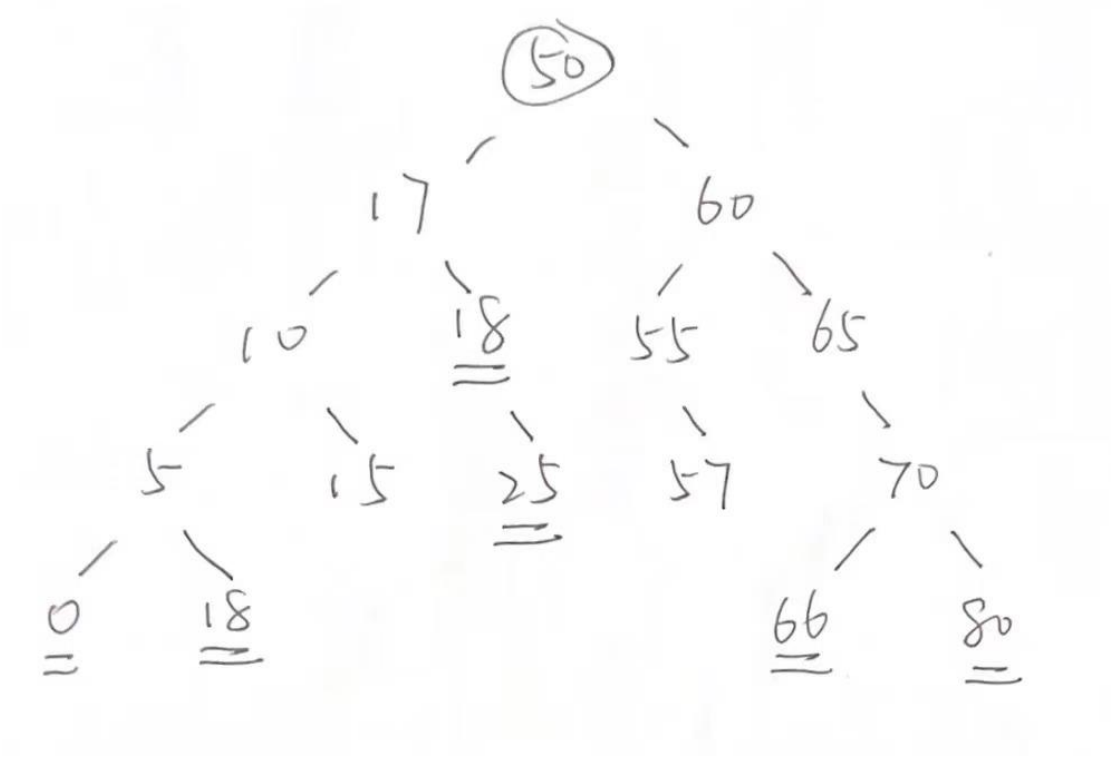
(i)

NLR:[50,17,10,5,15,60,55,57,65,70]

LNR:[5,10,15,17,50,55,57,60,65,70]

LRN:[5,15,10,17,57,55,70,65,60,50]

(ii)



(iii)

LNR:[0,5,8,10,15,17,18,25,50,55,57,60,65,66,70,80]

(iv)