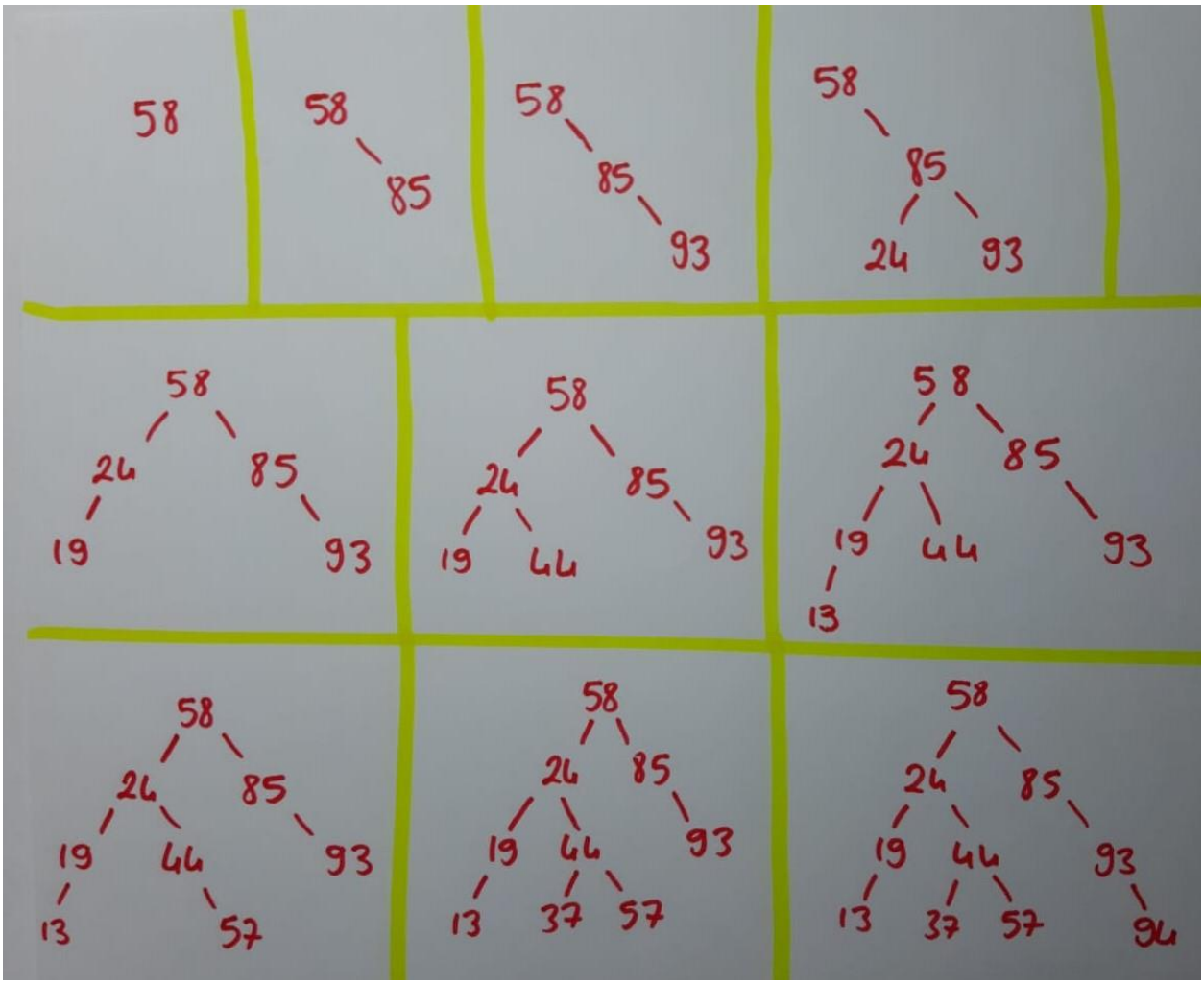
/**
* Author : Melike Demirci
* ID: 21702346
* Section : 2
* Assignment : 2
*/

Question 1

a) Insert 58, 85, 93, 24, 19, 44, 13, 57, 37, 94 into an empty binary search tree (BST) in the given order. Show the resulting BSTs after every insertion.



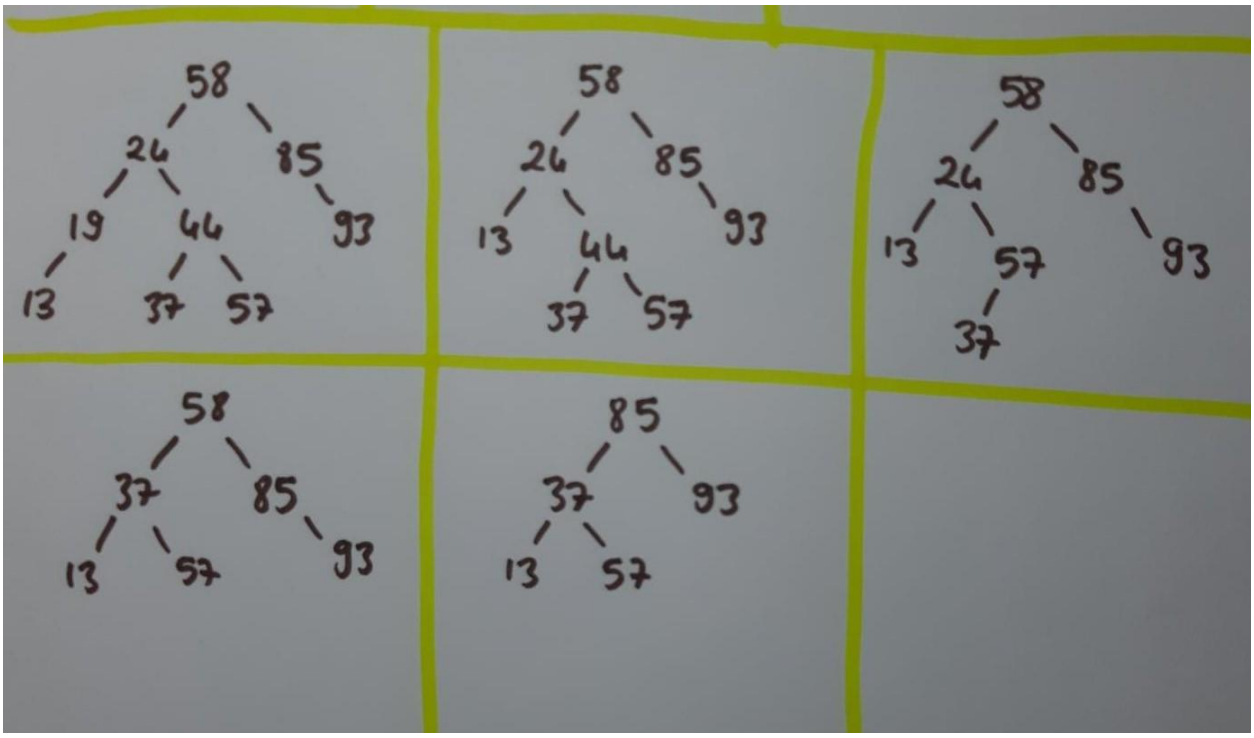
b) What are the preorder, inorder, and postorder traversals of the BST you have after?

Inorder => [13, 19, 24, 37, 44, 57, 58, 85, 93, 94]

Preorder => [58, 24, 19, 13, 44, 37, 57, 85, 93, 94]

Postorder => [13, 19, 24, 37, 44, 57, 85, 93, 94, 58]

c) Delete 94, 19, 44, 24, 58 from the BST you have after **(a)** in the given order. Show the resulting BSTs after every deletion



A) Train Functions

- **Recursive Train Function**

trainHelper (Pseudo Code)

```
trainHelper(){
   numUsedFeatures(numFeatures, usedFeatures);
   getFrequentLabel(labels, usedSamples, numSamples);
   getClassInfo(numSamples, labels, usedSamples, classCounts, numClasses);
   calculateEntropy(classCounts, numClasses);
   if(noFutureLeft){
     treePtr = new DecisionTreeNode(freqLabel,true)
   }
   else if(entropy <= 0.0){
     treePtr = new DecisionTreeNode(freqLabel,true)
   }
   else{
     //Calculate information gain for the available features
          numFeatures*O(calculateInformationGain)
      //Create new decision tree node
      //Create new bool arrays for new usedFeatures and usedSamples child nodes and fill
      2 for loop upper bound numSamples + 2 for loop upper bound numFeatures
     //call the function for the childs recursively
      trainHelper(for left child ptr);
      trainHelper(for right child ptr);
   }
}
```

Train helper function is a recursive function, it calls itself two times in the else part. In balanced binary trees insertion, deletion and search operations are proportional to height of the tree which equals to logn(n is the number of nodes). So balanced binary tree with height h needs O(logn) time for insertion, deletion and search.

However, decision trees doesn't have to be balanced.

- **Functions Related to Information Gain**

  double calculateInformationGain

  O(max(3* O(getClassInfo), numSamples, 3*O(calculateEntropy))

  double calculateEntropy

  depends on number of numClasses

  O(numClasses)

  void insertionSort

  O(n^2)

  void getClassInfo

  O(max(numSamples, numClasses))

- **Other Helper Functions**

  void train

  max(O(trainHelper),O(numSamples), O(numFeatures))

  bool numUsedFeatures

  O(numFeatures)

  int getFrequentLabel

  O(numSamples^2)