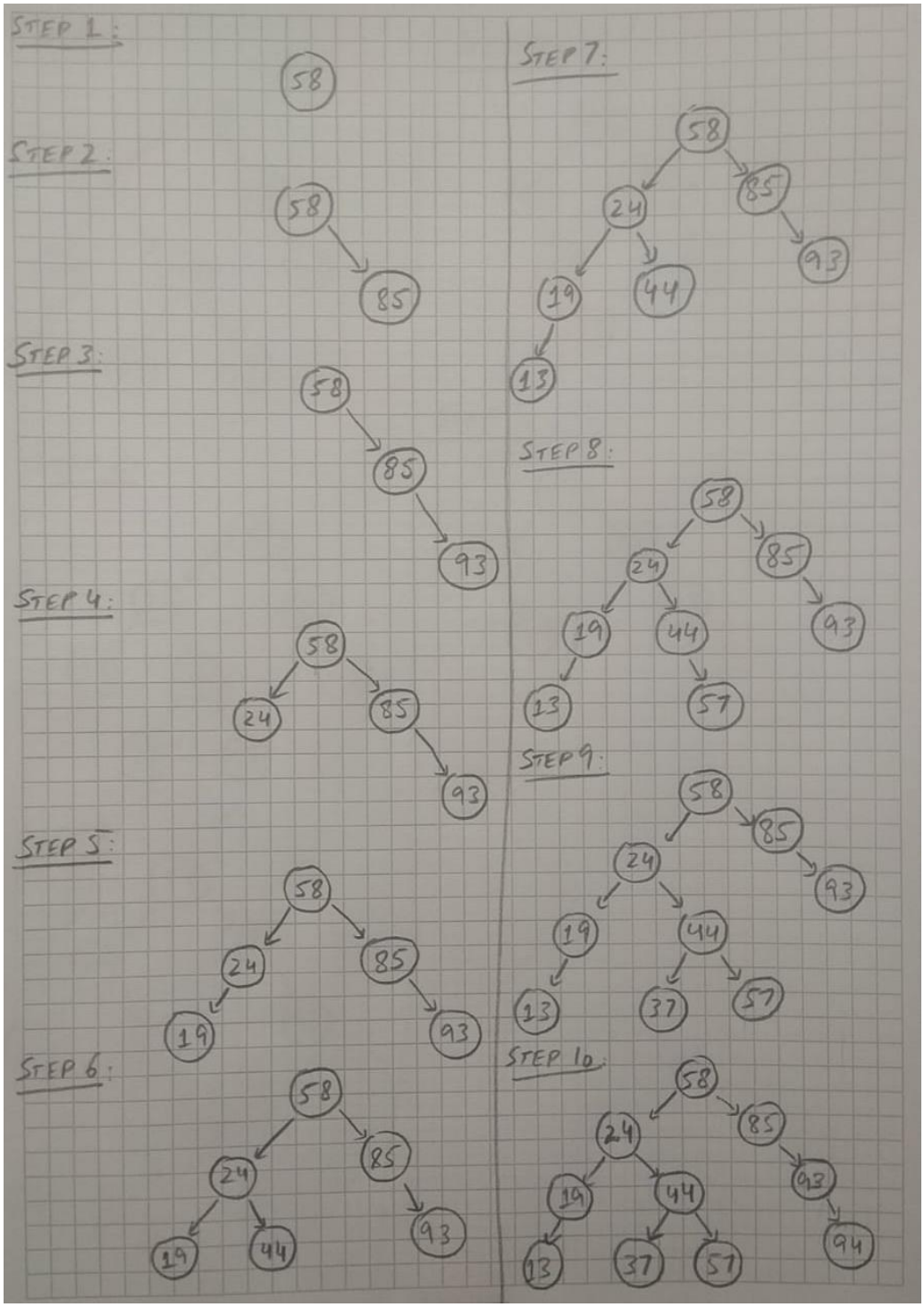
Abdul Basit Anees
21600659
CS 202-3

Homework 2

Question 1

a)



Abdul Basit Anees
21600659
CS 202-3

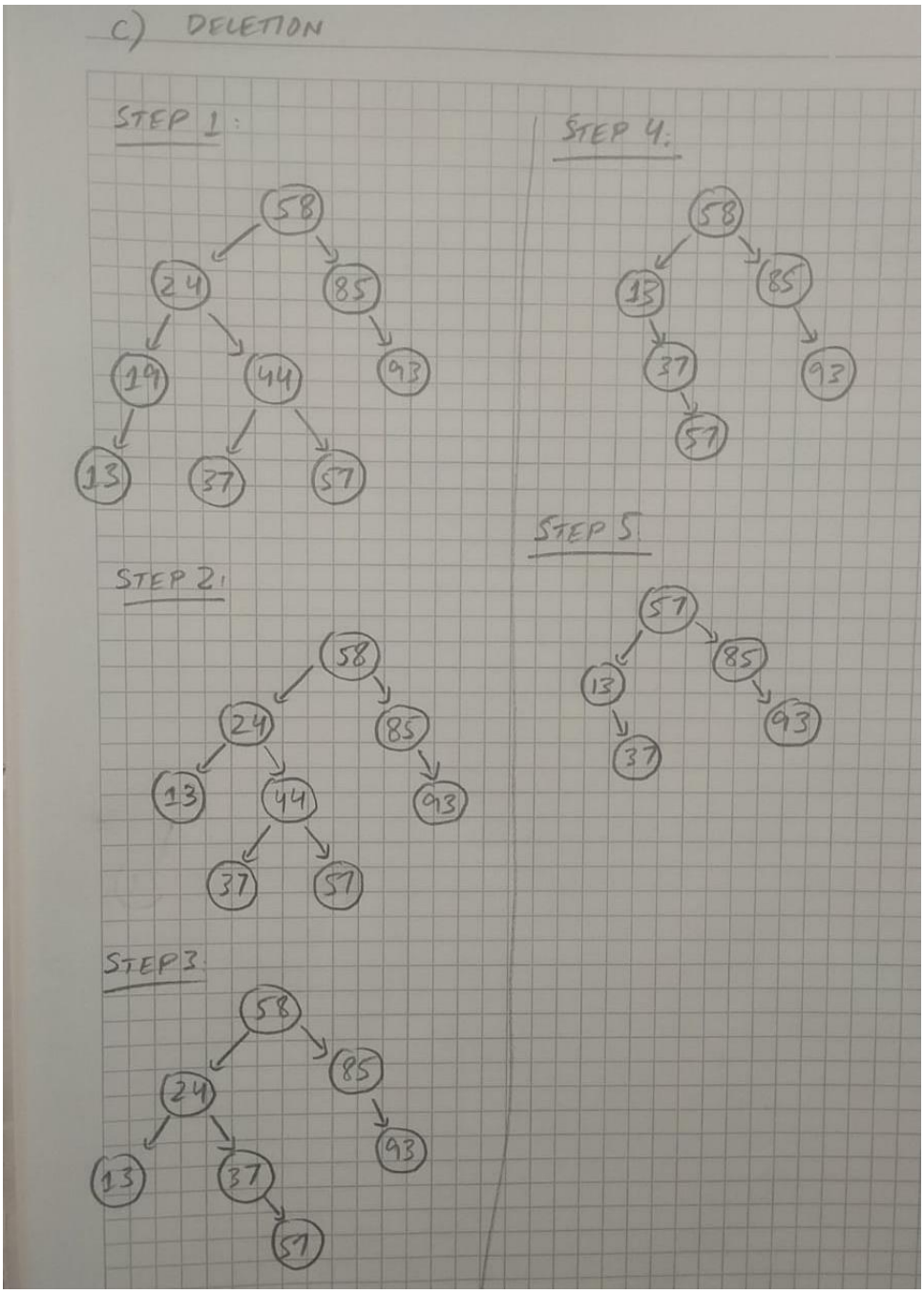
b)

Preorder: 58 24 19 13 44 37 57 85 93 94

Inorder: 13 19 24 44 37 57 58 85 93 94

Postorder: 13 19 37 57 44 24 94 93 85 58

c)



Abdul Basit Anees
21600659
CS 202-3

Question 3

To include the classes belonging to any integer, a kind of dictionary was implemented by using the function findClasses which finds the number and values of unique classes in the data. The explanations of different functions are as follows:

double calculateEntropy(const int* classCounts, const int numClasses)

This functions just uses the formula for entropy to calculate entropy and assumes 0 * log(0) = 0

double calculateInformationGain(const bool data, const int* labels, const int numSamples, const int numFeatures, const bool* usedSamples, const int featureId)**

This function calculates the parent entropy and then entropy of the children. The Information Gain is the difference of parent entropy and weighted children entropy.

void DecisionTree::train(const string fileName, const int numSamples, const int numFeatures)

This function loads the data from text file and passes the data into other train function.

void DecisionTree::train(const bool data, const int* labels, const int numSamples, const int numFeatures)**

This function passes the DecisionTreeNode root and data into the split function which modifies the root to form a tree.

void DecisionTree::split(DecisionTreeNode *&node, const bool data, const int* labels, const int numSamples, const int numFeatures, const bool* usedSamples, const bool* usedFeatures, int*& classes, int& numClasses);**

This function first checks if the node is pure. If yes then the function is returned with the class decision added to the node.

Then it checks if all features have been used in the branch. If yes, then function is returned with decision added to node made by the majority class.

Otherwise, the information gains are calculated for unused features and using the feature with max IG, the split function is recursively called to the child nodes.

int predict(const bool* data);

int predict1(const bool* data, DecisionTreeNode node);

predict1 is called in the predict function with the data and the root of the decision tree as inputs. Then, it returns the class decision if the node is a leaf node, otherwise recursively calls the predict1 function again depending on the feature used by the node to split.

The time complexity of this is O(log(n)) where n is the number of nodes.

Abdul Basit Anees
21600659
CS 202-3

double DecisionTree::test(const string fileName, const int numSamples)

This reads testData from the file and passed the data to other test function.

double DecisionTree::test(const bool data, const int* labels, const int numSamples)**

This used the predict function to predict the class of each row of the test data and then calculates the accuracy of the decision true using the predictions.

void deleteRecursively(DecisionTreeNode *node)

This function deletes a node if it is leaf node, otherwise deletes the left and right children and then makes it a leaf node to delete it. The time complexity of deletion is O(n) where n is the number of nodes.