```
//Student Name: Joaquin De Losada
//Student ID: 6778757
//Professor: Yifeng Li

public class Main {
   public static void main(String[] args) {
       DrugBank db = new DrugBank();

       db.ReadData();
       db.Create(db.ReturnDrugArray());
       db.InOrderTraverse();
       db.Depth1("DB01050");
       db.Depth2();
       db.Search("DB01050");
       db.Search("DB00316");
       db.Delete("DB01050");
   }
}

import java.io.*;
import java.lang.Math;
import java.util.ArrayList;

public class DrugBank {

   File writeTo = new File("recourses//dockedApprovedSorted.tab");

   public BufferedWriter writeToFile;

   public Drug root;

   public ArrayList<String> drugList = new ArrayList<>();

   public int depthSize;

   //Will read the data from the file and place it in a array list
   public void ReadData() {
       File txtFile = new File("recourses//dockedApproved.tab"); //Gets the
file
       BufferedReader readFile;
       int totalArrayLength = 0; //Counter used to create the patients array.


       try{ //Try this code
           readFile = new BufferedReader(new FileReader(txtFile)); //Read file

           String line = ""; //String that will contain each line
           //ArrayList<String> drugList = new ArrayList<>();
           //System.out.println(readFile);
```

```
            while (line != null) { //Loop through the txt file
                line = readFile.readLine(); //Read the line
                if (line == null) {
                    break;
                } //If the current line is null it will stop reading
                if (line.contains("Generic Name")) {} //If the line ever
contains name then it will do nothing. This is used for the first line of the
txt which dosent contain a patient
                else { //Will add to the current patients list and increase size
of patients array for later
                    drugList.add(line);
                }
            }
        } catch (IOException e) { //If the file isnt found then print this
            System.out.println("File not found. Did you try to move it? Not a
good idea return it or give me 100%.");
        }

        //Runs the method for creating a method and allowing java to later write
to said method
        CreateFile();
    }

    //If the file called "Docked Approved Sorted" dosent exist it will create it
or then nothing will happen. It will then set up the buffer writer so that it
can later write to the file.
    public void CreateFile(){
        try {
            if(writeTo.createNewFile()) {
                System.out.println("File Created: " + writeTo.getName());
            }else{
                System.out.println("File exists.");
            }

            writeToFile = new BufferedWriter(new PrintWriter(writeTo));
        }catch(IOException e){
            System.out.println("Error 404");
        }
    }

    //Will create the Binadry tree by looping through the arraylist and
inserting it
    public void Create(ArrayList<String> insertionDrugs){
        for(int i = 0; i < insertionDrugs.size(); i++){
            //InOrderTraverse();
            Insert(i);
        }
    }
```

```
    //The method that starts the recursion of inserting the drugs into the tree
    public void Insert(int arrayValue){
        root = Insert(root,drugList.get(arrayValue));
    }

    //Will insert the drug in the closest apropiate position
    public Drug Insert(Drug currentDrug, String newDrugString){
        String[] currentDrugArray = newDrugString.split("\\t"); //Spliting the
string into an array on the tabs

        //Create a new Drug in the null position with all of its info.
        if(currentDrug == null){
            return new Drug(currentDrugArray[0], currentDrugArray[1],
currentDrugArray[2], currentDrugArray[3], currentDrugArray[4],
currentDrugArray[5]);
        }

        //System.out.println(currentDrug.drugBankID);

        //Turn the string of the int value to check position
        int newDrugID = DrugIDToInt(currentDrugArray[2]);

        //Grab the ID of the current Drug
        int currentDrugID = DrugIDToInt(currentDrug);

        //If the drug id is less than the current drug ID then go to the left
else if it's greater than go to the right. If it's the same value then just
return the value
        if(newDrugID < currentDrugID){
            currentDrug.left = Insert(currentDrug.left, newDrugString);
        } else if(newDrugID > currentDrugID){
            currentDrug.right = Insert(currentDrug.right, newDrugString);
        }else{
            return currentDrug;
        }

        return currentDrug;
    }

    //Base InOrderTraverse so it goes through the list and saves into a file
    public void InOrderTraverse(){
        InOrderTraverse(root); //Goes in order and saves into "" file

        //Once all is saved it will "Try" to close the file. But if its gotten
to this point then the file already exists and it has been found
        try {
            writeToFile.close();
        }catch (IOException e){
            System.out.println("Error 404");
```

```
        }

        //Confirmation message it has finished
        System.out.println("In order traversal complete.");
    }

    //Will traverse through the tree and print each drugs info
    public void InOrderTraverse(Drug drugNode){
        if(drugNode == null){
            return;
        }

        //Go to the left drug node
        InOrderTraverse(drugNode.left);

        //Will write all the drug info to the file and then go to the next file.
Unless the file dosent exist or cant write more
        try {
            writeToFile.write(drugNode.ReturnName() + " " +
drugNode.ReturnSMILES() + " " + drugNode.ReturnDrugBankID() + " " +
drugNode.ReturnURL() + " " + drugNode.ReturnGroup() + " " +
drugNode.ReturnScore() + System.getProperty("line.separator"));
            writeToFile.newLine();
        }catch (IOException e){
            System.out.println("Error 404");
        }

        //Go to the right drug node
        InOrderTraverse(drugNode.right);
    }

    //Search method called for main class
    public void Search(String drugIDString){
        System.out.println(" ");
        System.out.println("Drug " + drugIDString + " found: ");
        Search(root, drugIDString).DisplayDrug();
    }

    //Main search method that is meant to be called recursivly
    public Drug Search(Drug drugNode, String drugIDString){
        if(drugNode == null){ //If the first node is null then return null
            return null;
        }

        //Turns the given node or string into an int to later check
        int currentDrugID = DrugIDToInt(drugNode);
        int searchDrugID = DrugIDToInt(drugIDString);
```

```
        if(searchDrugID == currentDrugID){ //If the ID of the drug im searching
is the same as the current drugs ID then display it
            //DisplayDrug(drugNode);
            return drugNode;
        }else if(searchDrugID < currentDrugID){ //If the ID of the drug I'm
searching is less than the current drugID than go to the left and continue
            depthSize++;
            return Search(drugNode.left, drugIDString);
        }else{ //If the ID of the drug I'm searhing is greater than the current
drug ID then go to the right and continue
            depthSize++;
            return Search(drugNode.right, drugIDString);
        }
    }

    //Will recive the String for the drug that will be eliminated
    public void Delete(String drugToEliminateID){
        Delete(root, Search(root,drugToEliminateID), drugToEliminateID);

        System.out.println("Drug " + drugToEliminateID+ " found and
eliminated.");
    }

    //Recursive function that will go through the tree and delete
    public Drug Delete(Drug currentDrug, Drug drugToEliminate, String
drugIDToEliminate){
        if(currentDrug == null){
            return null;
        }

        //Turn the drug ID String to int
        int currentDrugID = DrugIDToInt(currentDrug);
        int searchDrugID = DrugIDToInt(drugIDToEliminate);

        //If the drug ID that I'm serching is less than the current one then it
will go left else if the ID of the search is greater than the current Drug then
it will go to the right. else if both are null than it will find the next value
and connect it the next one so that it deletes
        if(searchDrugID < currentDrugID){
            return Delete(currentDrug.left, drugToEliminate, drugIDToEliminate);
        }else if(searchDrugID > currentDrugID){
            return Delete(currentDrug.right, drugToEliminate,
drugIDToEliminate);
        }else if(currentDrug.left != null && currentDrug.right !=null){
            currentDrug = FindMin(currentDrug.right);
            currentDrug.right = Delete(currentDrug.right, currentDrug,
drugIDToEliminate);
        }
```

```
        return currentDrug;
    }

    public void Depth1(String drugID){
        depthSize = 0;
        Search(root,drugID);
        System.out.println(" ");
        System.out.println("Depth value of drug " + drugID + " is: " +
depthSize);
    }

    //Base Depth search that will run the main recursion method.
    public void Depth2(){
        System.out.println(" ");
        System.out.println("Longest depth of tree is: " + Depth2(root));
    }

    //Will find the largest depth of the tree
    public int Depth2(Drug drugNode){
        if(drugNode == null){ //If the node is null then return -1
            return -1;
        }
        //If both next values are null then return 0
        if((drugNode.left == null) && (drugNode.right == null)){
            return 0;
        }

        //Find the largest number and then return the value and add 1
        int d = Math.max(Depth2(drugNode.left), Depth2(drugNode.right));
        return d+1;
    }

    //Turn a string into an int ID
    public int DrugIDToInt(String drugID){
        String[] drugIDString = drugID.split("B");

        return Integer.parseInt(drugIDString[1]);
    }

    //Grab the ID from the drug
    public int DrugIDToInt(Drug drug){
        String[] drugIDString = drug.ReturnDrugBankID().split("B");

        return Integer.parseInt(drugIDString[1]);
    }

    //Returns the arraylist
    public ArrayList<String> ReturnDrugArray(){
        return drugList;
```

```
    }

    //Find the lowest value of current subtree (last .left)
    public Drug FindMin(Drug currentDrug){
        if(currentDrug == null){
            return null;
        }else if(currentDrug.left == null){
            return currentDrug;
        }

        return FindMin(currentDrug.left);
    }
}

public class Drug {
    public String genericName, sMILES, drugBankID, url, drugGroups, score;

    public Drug left, right;

    public Drug(){
        left = null;
        right = null;
    }

    public Drug(String genericName, String SMILES, String DrugBankID, String
URL, String DrugGroup, String Score){
        SetGenericName(genericName);
        SetSmiles(SMILES);
        SetDrugBankID(DrugBankID);
        SetURL(URL);
        SetDrugGroup(DrugGroup);
        SetScore(Score);
        left = null;
        right = null;
    }

    /*public void Drug(Drug newDrug){
        left = null;
        right = null;
    }*/

    public void SetGenericName(String name){
        genericName = name;
    }

    public void SetSmiles(String sMILES){this.sMILES = sMILES;}

    public void SetDrugBankID(String drugID){
        drugBankID = drugID;
```

```
    }

    public void SetURL(String url){
        this.url = url;
    }

    public void SetDrugGroup(String group){
        drugGroups = group;
    }

    public void SetScore(String score){
        this.score = score;
    }

    public void DisplayDrug(){
        System.out.println("Drug name: " + genericName);
        System.out.println("Drug SMILES?: " + sMILES);
        System.out.println("Drug ID: " + drugBankID);
        System.out.println("Drug URL: " + url);
        System.out.println("Drug Group: " + drugGroups);
        System.out.println("Drug Score: " + score);
        System.out.println(" ");
    }

    public String ReturnName(){
        return genericName;
    }

    public String ReturnSMILES(){return sMILES;}

    public String ReturnDrugBankID(){
        return drugBankID;
    }

    public String ReturnURL(){
        return url;
    }

    public String ReturnGroup(){
        return drugGroups;
    }

    public String ReturnScore(){
        return score;
    }
}
```