GEBZE TECHNICAL UNIVERSITY

CSE 222
DATA STRUCTURES AND ALGORITHMS

HOMEWORK 5
REPORT

OGUZ MUTLU
1801042624

1. DETAILED SYSTEM REQUIREMENTS

This homework

- For Input of the system should accept a .txt file as input. Each row in the file represents a data point, and the data is split into categories by the ";" character. Each column in the file can have an arbitrary number of unique values. In each part in this assignment accept different inputs from the user. For example B,C,D parts we have to use console input from taking an input from the user
- Parsing: The system should parse the data from the .txt file and represent it as a tree structure. The tree structure should not necessarily be a binary tree, but it should be a tree structure. And parsing txt file we have to parse txt into 2d string array which sizes grows dynamically
- Printing: The system should be able to print the parsed tree structure in a JTree and JFrame format. This could be as a visual tree diagram, or as a list of nodes and their relationships to one another.
- Error Handling: The system should handle errors gracefully. If the input file is not formatted correctly, the system should provide informative error messages to the user. If the file is not found or there is a problem reading the file, the system should also provide informative error messages.

2.PROBLEM SOLUTION APPROACH AND IMPLEMENTATION

Tree() this function creates a tree from the given 2d string, This function use iterative method behind its idea is,

- first we create a root node of the tree which is root then this function check first row of the array in every splitted character node.add() method invokes by method after first column was written other columns writtten to that node
- I move root reference to the child node and after this reference move iteration (we assign root = child) and then we invoke add function on the root node. It adds every column by root iteration with help of this function

insertTheTree() this function looks if given node is insertable to the tree, tree already include given node data it must skip that node this method check that situation with helper isInclude method

moveTreeObject() if new node is already added to the tree then this function iterates through the children to reach the data node inside the tree root of the tree node data node to insert or iteratively moveif data node object found in the tree returns founded node reference

BFS()

- First I accessed the child nodes starting from root, then I wrote a for loop that will visit each child nodes of root first.
- Within this for loop, I assigned child node I visited to a queue, then I visited the 2nd and 3rd child nodes if exists,
- starting with the first child I get childs child nodes , and assigned these values to the queue in that order. While doing these, at the same time,
- I checked whether the value I was looking for and the value I would assign to the queue at the moment matched, and if the result was correct, I finished the process.

DFSRecursion()

I use stack data structure for this method also write recursive but I choose stack solution in the homework you can also check the code and recursive part.

- Recursively searches the subtree rooted at the given node using the Depth-First Search algorithm. The method visits each node in the subtree by first exploring the rightmost branch until it reaches a leaf node, then backtracks and explores the next unvisited branch.
- The method stops the traversal and returns true when it finds a node with user object equal to the given input. If no such node is found, the method returns false.

DFSStack()

I use stack data structure for this method also write recursive but I choose stack solution in the homework you can also check the code and recursive part.

- Searches the tree using the Depth-First Search algorithm with a stack.
- Starting from the root node, the method traverses the tree by visiting each node's children first before moving on to the next sibling.
- The method stops the traversal and returns true when it finds a node with user object equal to the given input. If no such node is found the method returns false.

PostOrderTraversal()

- Recursively traverses the tree using post-order traversal and searches for the given user input.
- The method starts by visiting the leftmost child, then its siblings, and then the parent.
- If the given user input is found in the tree, it prints the steps taken and returns true.
- If the given user input is not found in the tree, it prints the steps taken and returns false.

move()

This function includes 4 methods as helper

- First I invoke c**reateNode()** method for create source nodes reference, I keep new tree for source path of the node then if last node has child node I take all last node reference and remove them

- -then our source path is arrange properly then we invoke add() function add function include 1 important function inside whichh is **iterateRoot()** method

  this methods usage purpose

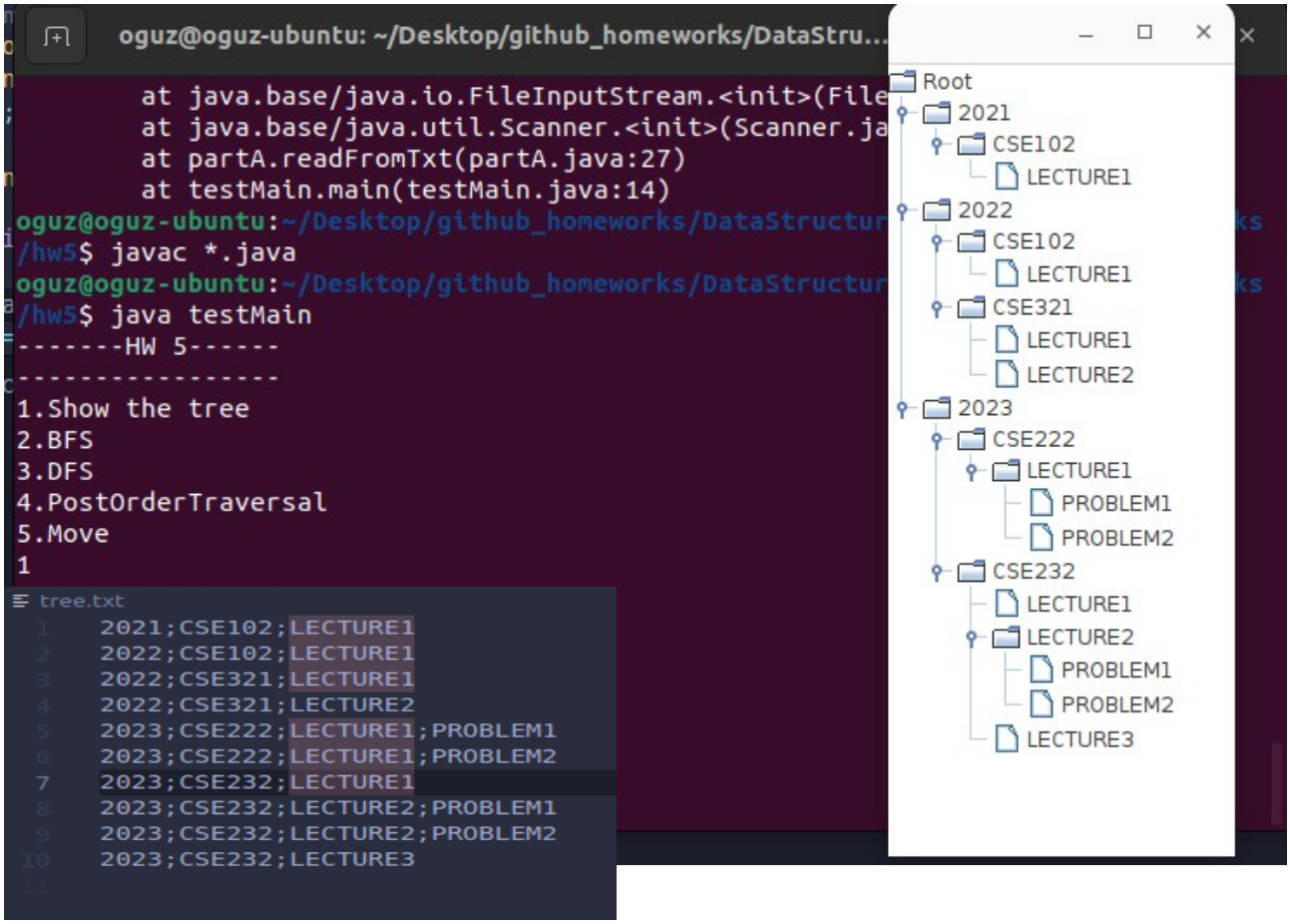
  - Iterates over a given root node to find a destination node represented by a string of comma-separated values.
  - If the destination node is found, it returns the node. Otherwise, it returns null.
  - If the destination node is not found, it inserts a new node to the tree and returns the newly created node.

After iteration done our reference which first assigned to the root is now assigned to the destination path in add function we arrange overwritten issue(let call our reference is tempReference), tempReference shows us the destination year but if there are same in the sourcePath and destination year we have to handle that issue I write simple check function iterates all child nodes of two part and then if exactly same nodes are found then I wrote to the console “overwritten” if not I add source path into the new destination path.

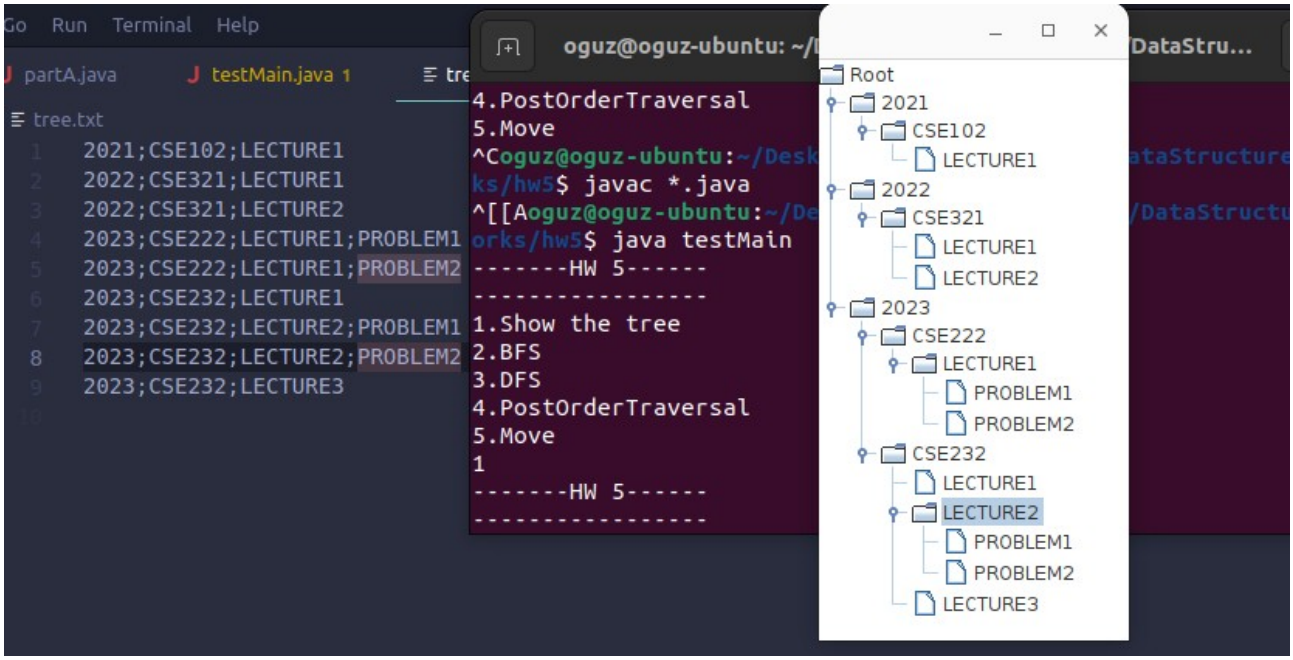
- Other method is **cleanTree()** method this method only deletes years which has not any child anymore

3.TEST CASES AND RESULTS

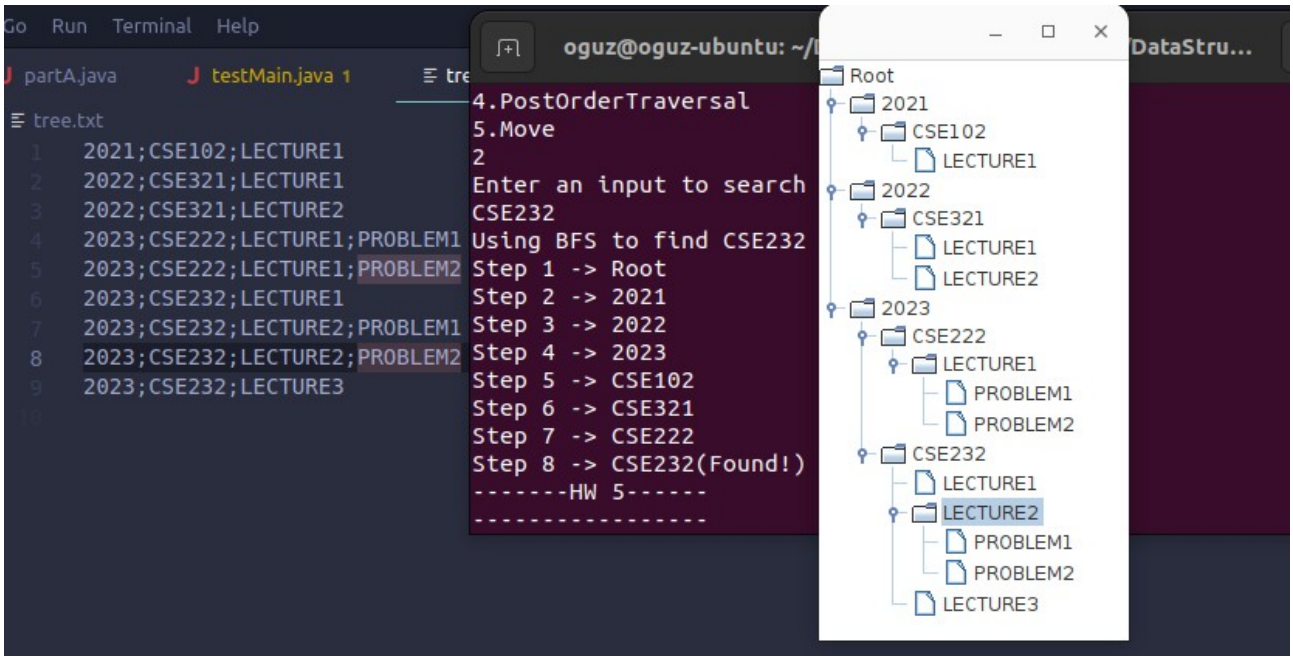
1. Showing the Tree Structure Readed From the Text



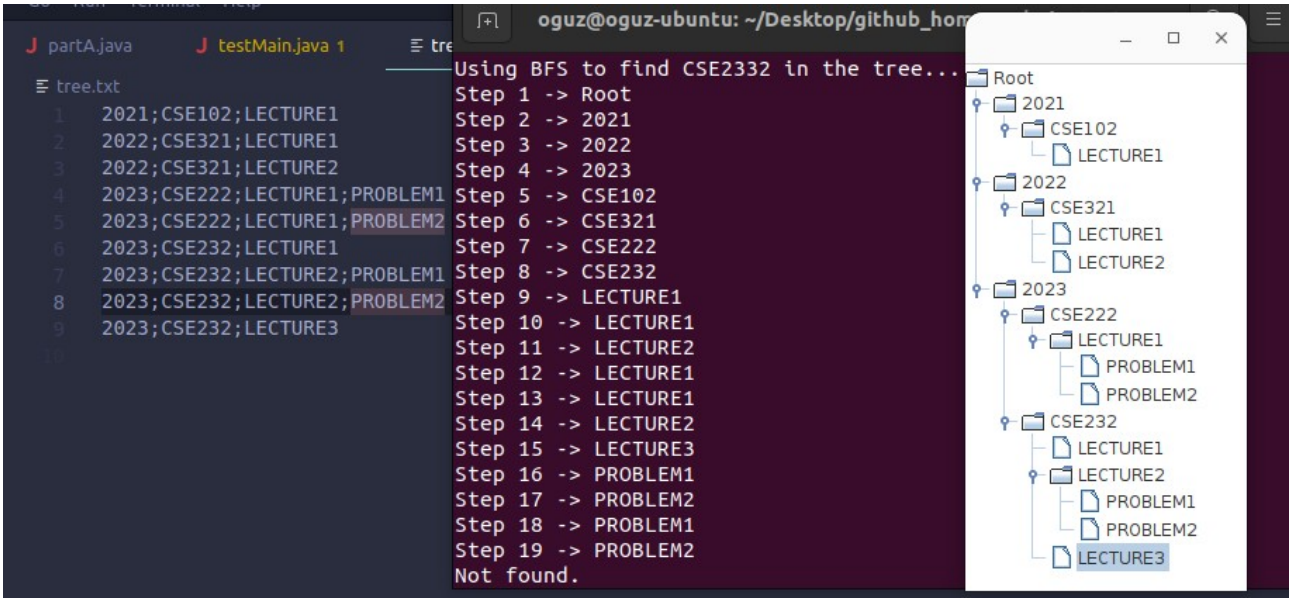
- Assingment example is in the below.



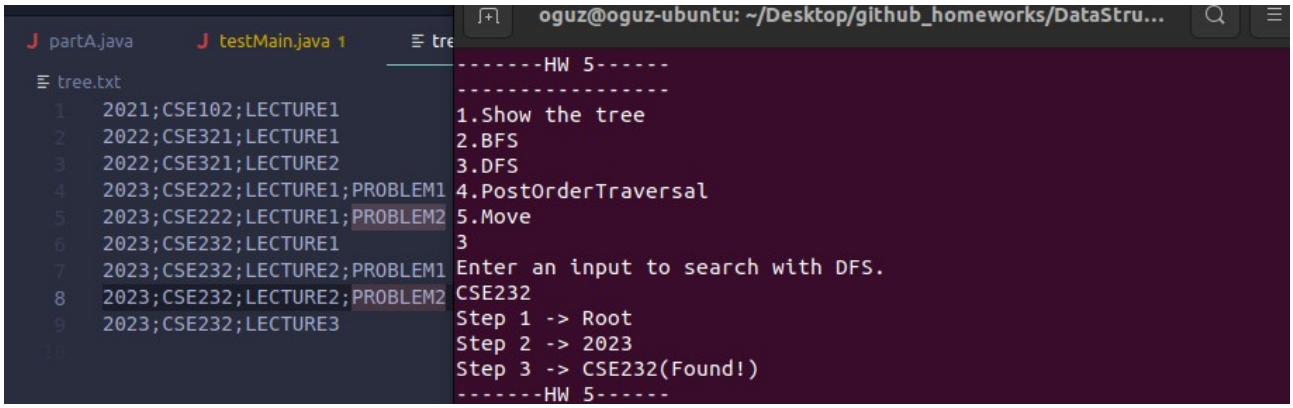
2.BFS Results

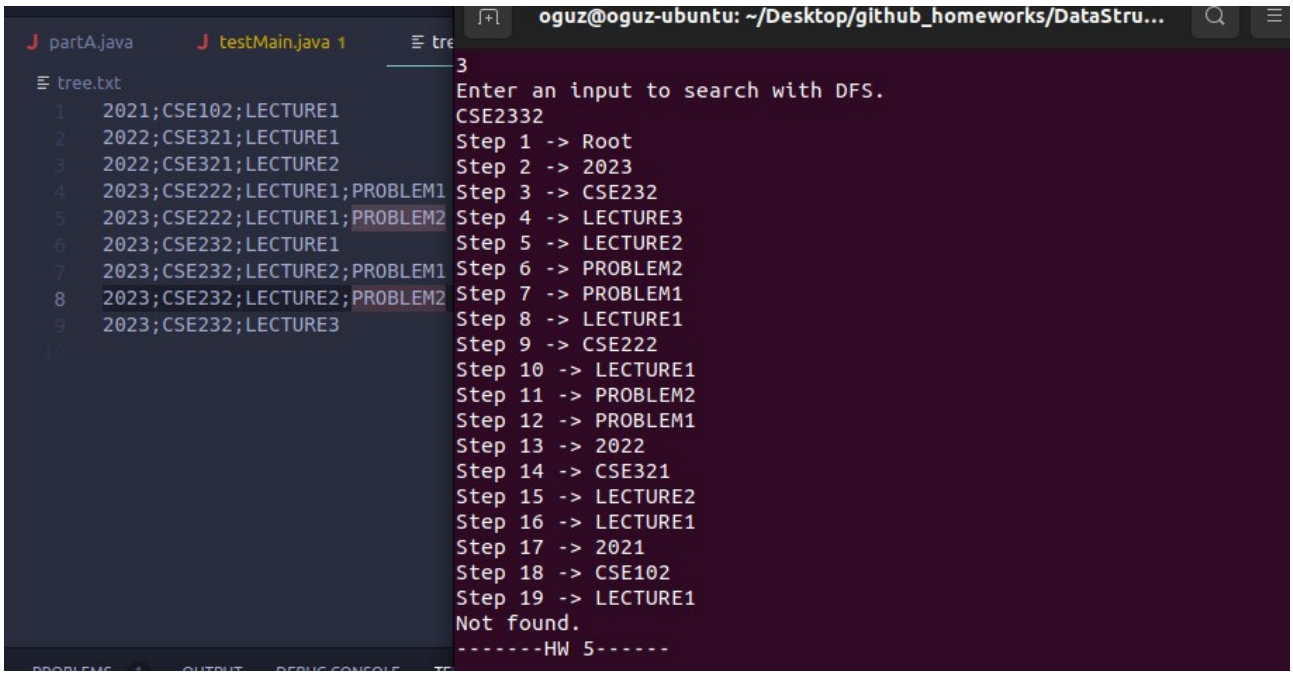


- Not found example

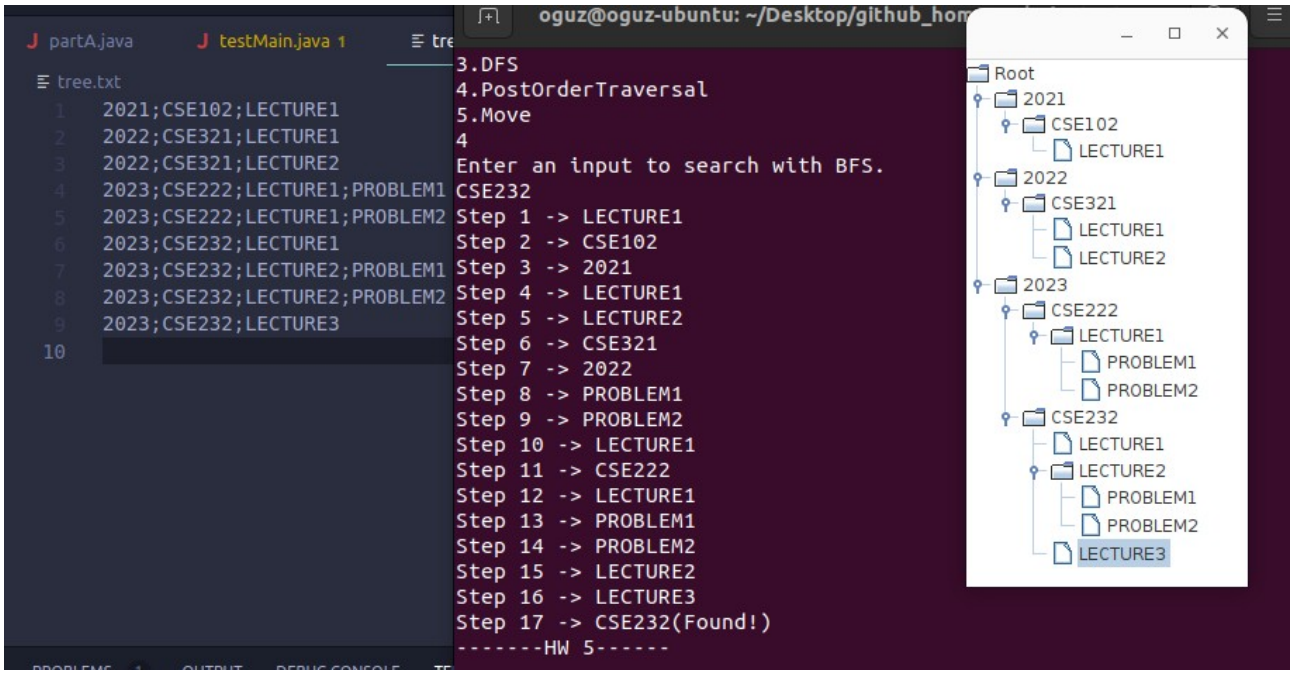


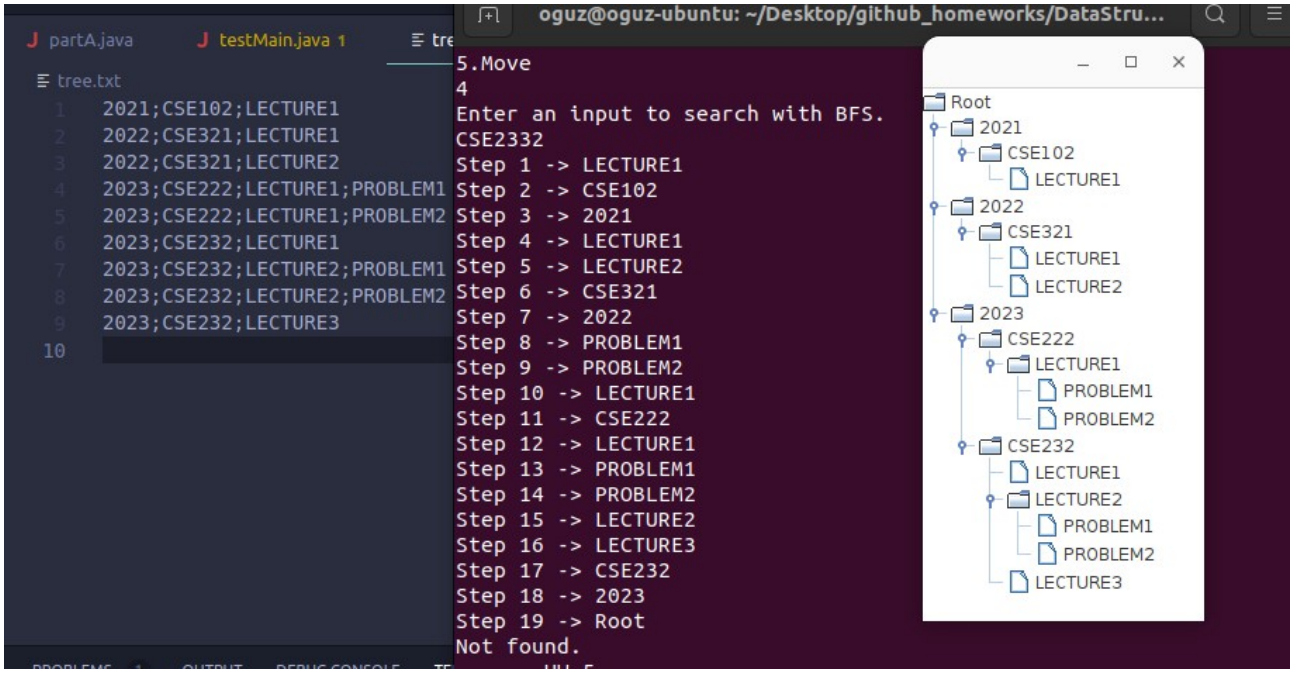
3.DFS Results



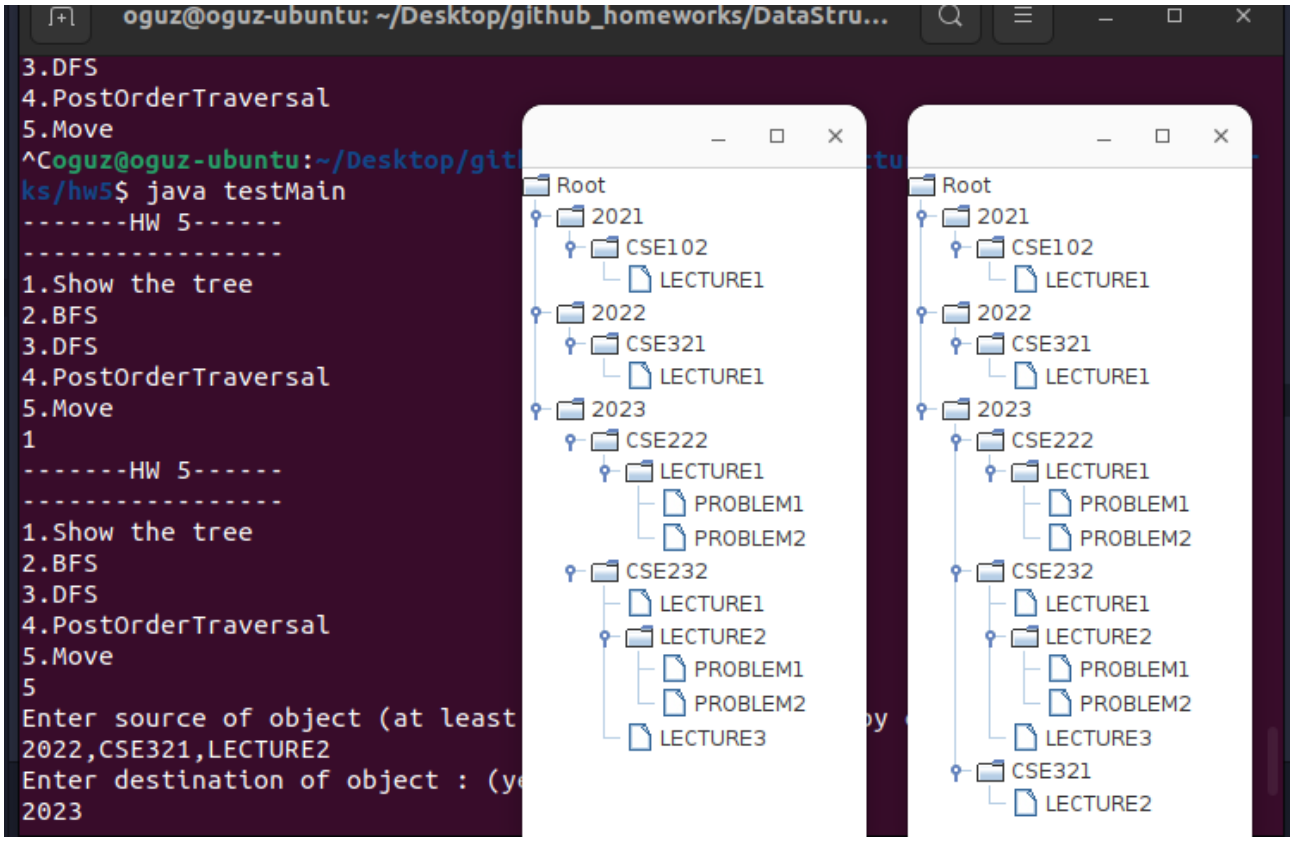


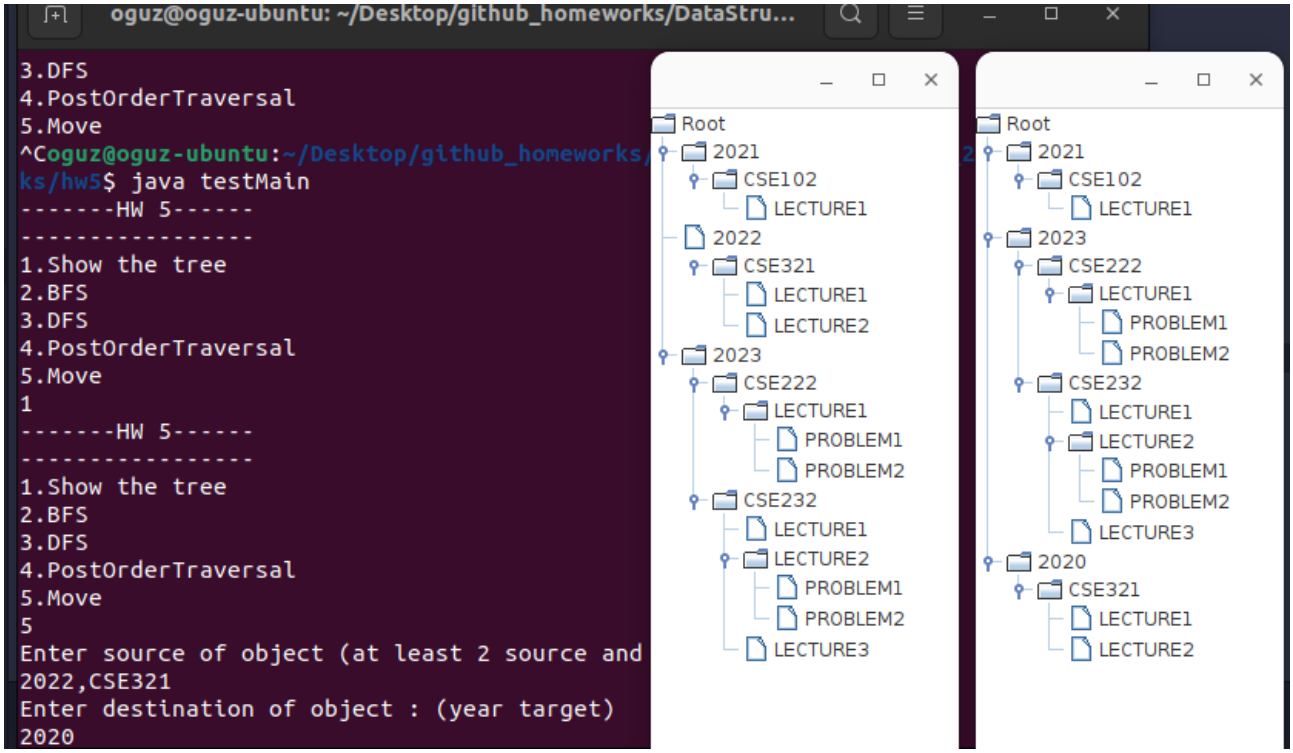
4.PostOrderTravelsal Results



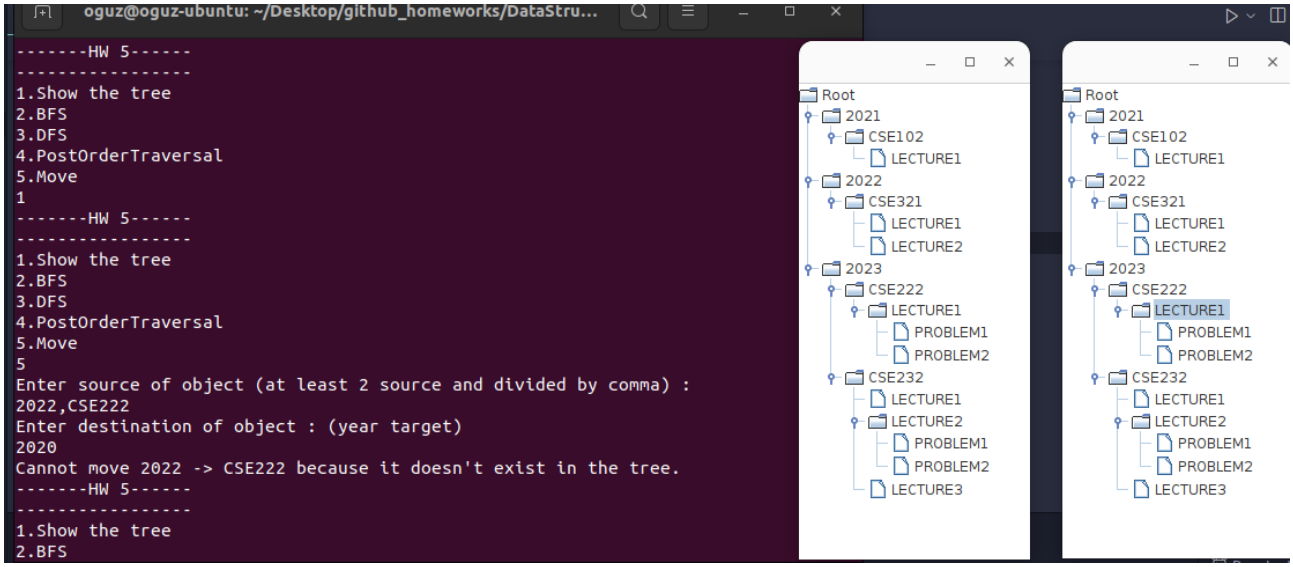


5.Move Results

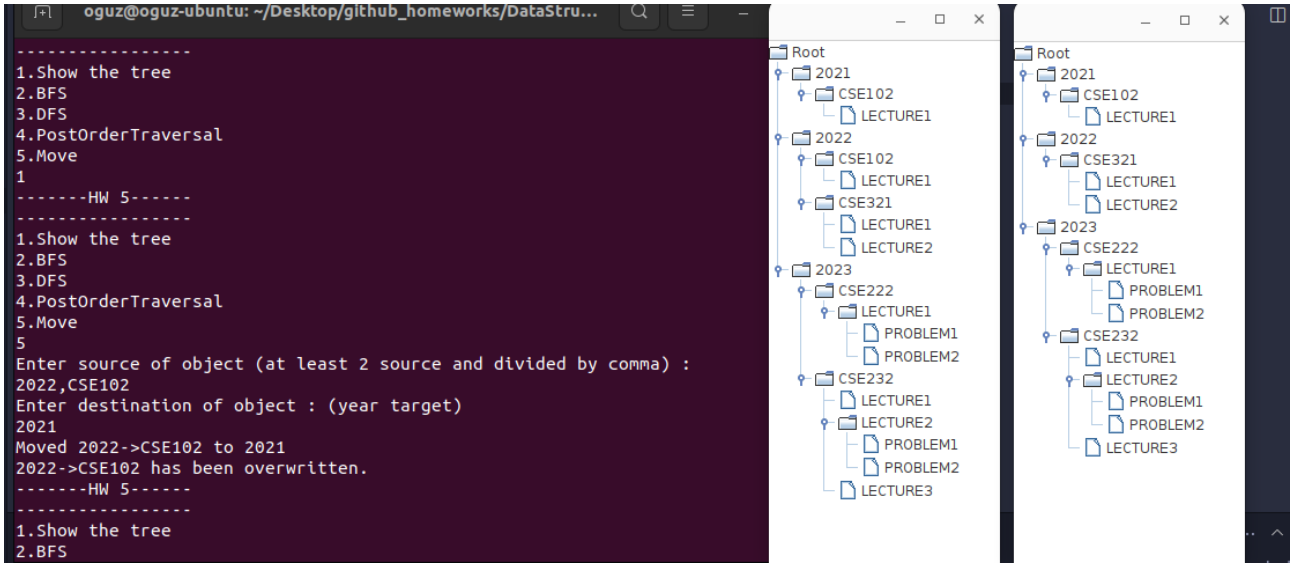




Writing to a console of not existing node

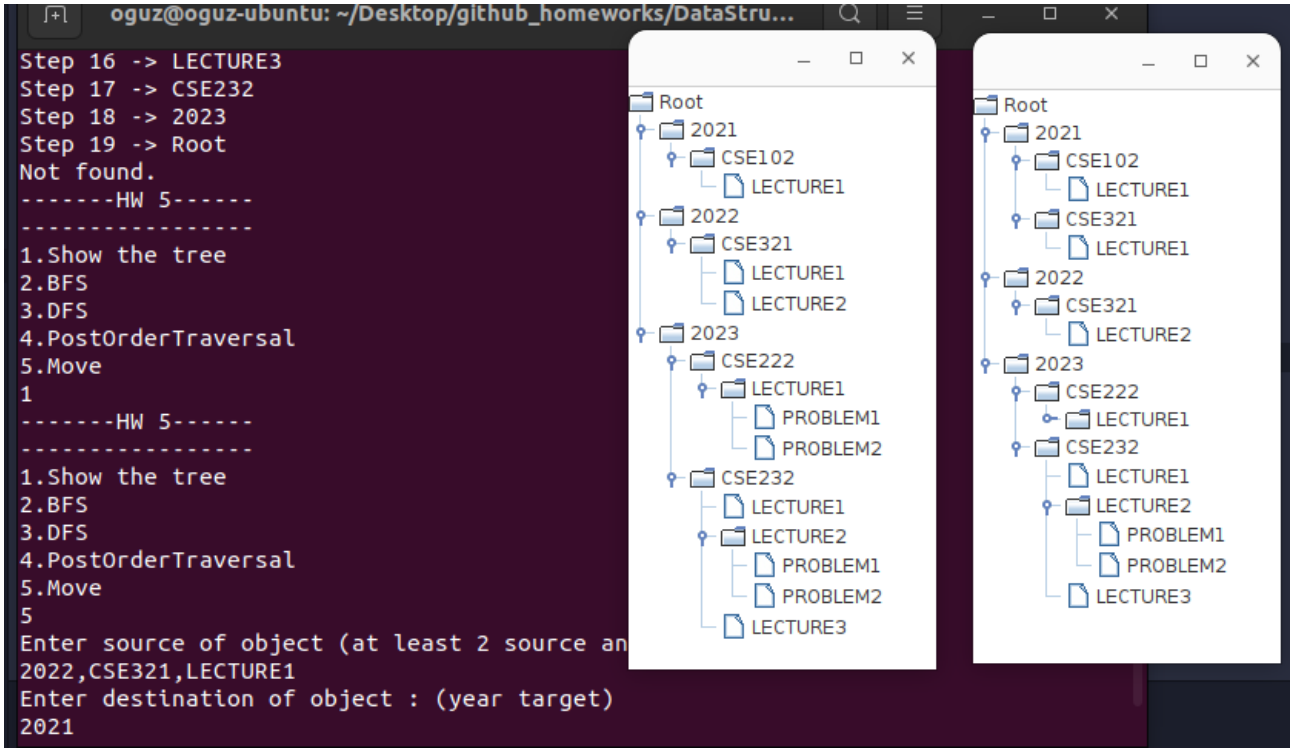


writing to a console of overwriting node



Test Case 1

- Move node to another year **PASS**



Test Case 2

- Enter Invalid Entry when moving, **PASS**
- Enter less than 2 parsed string in source input **PASS**
- Enter non numeric string value in destination input **PASS**
- Invalid entry for menu **PASS**