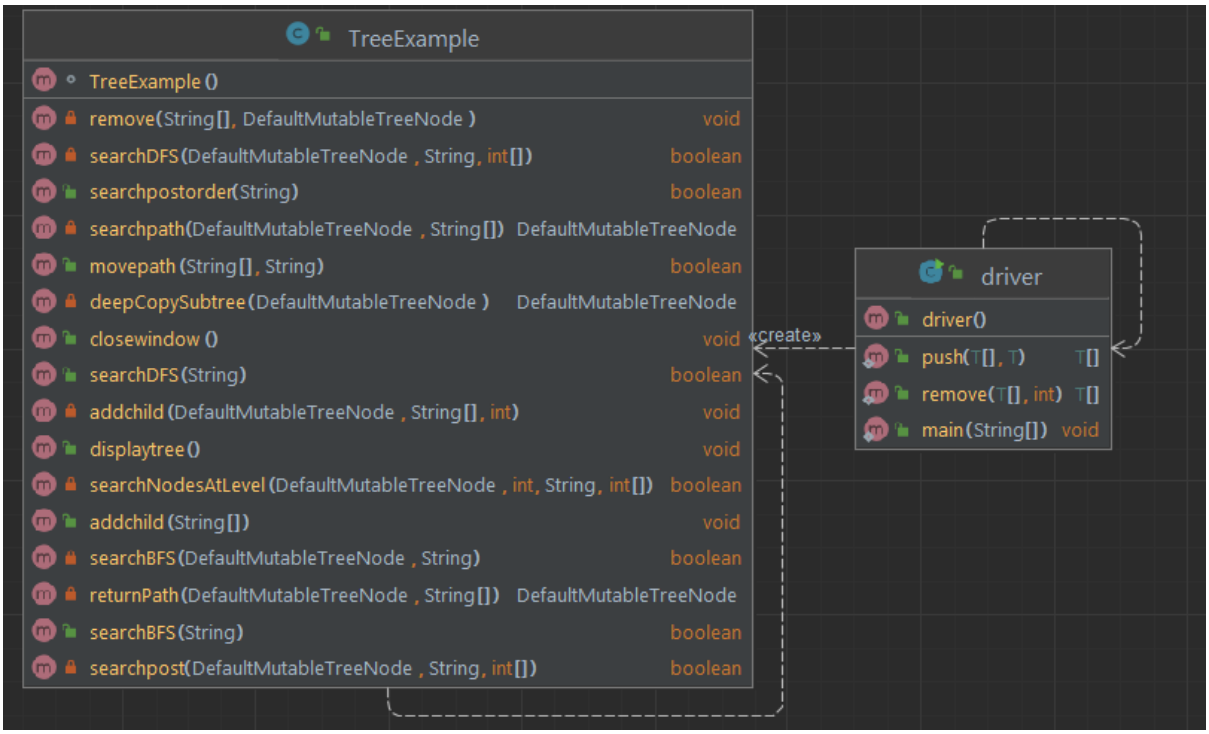
GIT Department of Computer Engineering
CSE 222/505 - Spring 2022
Homework # Report

HAMZA KONAÇ
210104004202

1. SYSTEM REQUIREMENTS

In this assignment want us create a java tree.for this assignment we must read file and add line to tree. After adding operation we must implement 3 differen tsearch algoirtm and move operation that move path to a distance node. For all of this operation we just need a TreeExample class.

2. USE CASE AND CLASS DIAGRAMS



3. OTHER DIAGRAMS

Don't required for this assignment

4. PROBLEM SOLUTION APPROACH

Creating tree ->

I have taken line from file and split it to by ; sign nad return a string array . I have do this all lines and add this to tree. For inserting Actuall every line is a path for tree. In adding function I have a root. Every level of root equal to index of the string array that crated by line which taken from list.if a index value that not in tree I create a node and insert it to parent of node.if index value is exist in tree I don't perform insertion just skip node

BFS Search->

BFS search algorithms search every level one by one . for do this I have implemented 2 function . first one has loop that increase by root level number. Second one is take level

value and item that will be search. Ins second function level ıs equal to serching level I wirte step and node values

DFS Search ->

DFS search created by recursively.private searchDFS method take root and searching item and call recursivelly until root be null. By using do this I have created DFS search. Befor calling reecursive for every child of node I have print steps screen and check if node is equa to serching item . if is equal I have return true for searching item

Post-Order Search ->

Post-Order search created by recursively.private searchpostorder method take root and searching item and call recursivelly until root be null. By using do this I have created postorder search. After calling recursive for every child of node I have print steps screen and check if node is equal to serching item . if it is equal I have return true for searching item

Move operation->

For moving firstly I have checked path is exist in tree.if exist have checked destination node.If move path exist in destination node I have removed subtree from source node. If not exist in destination node I have create new subtree that contain path nodes then insert it to destination node.

For remove node from source node I traverse to last node in the path given from by user. After that I remove it from the tree and go up to tree.during go up if parent node don't have children I remove it from the tree . But if parent node has child I break the loop and finish the remove operation.

For take path I create node by index in the path array that contain node names. In the last index I have create subtree and connect all of this node eachother . End of the this operation I have return first node of the new tree and insert it to destination. İmportant point in this operation if destination childreen is the same as move path I have replaced them(6^{th} output consider this point!!)

5. TEST CASES

->

tree.txt:

2021;CSE102;LECTURE1

2022;CSE321;LECTURE1

2022;CSE321;LECTURE2

2023;CSE222;LECTURE1;PROBLEM1

2023;CSE222;LECTURE1;PROBLEM2

2023;CSE232;LECTURE1

2023;CSE232;LECTURE2;PROBLEM1

2023;CSE232;LECTURE2;PROBLEM2

2023;CSE232;LECTURE3

2023;CSE321;LECTURE1

202;CSE102;LECTURE3

FOR BFS:

CSE224

CSE 232

FOR DFS:

CSE232

CSE 212

FOR Post-Order:

CSE232

CSE 212

FOR move:

2023->CSE232 to 2021

2021->CSE102->LECTURE1 to 2023

2023->CSE321->LECTURE to 2022

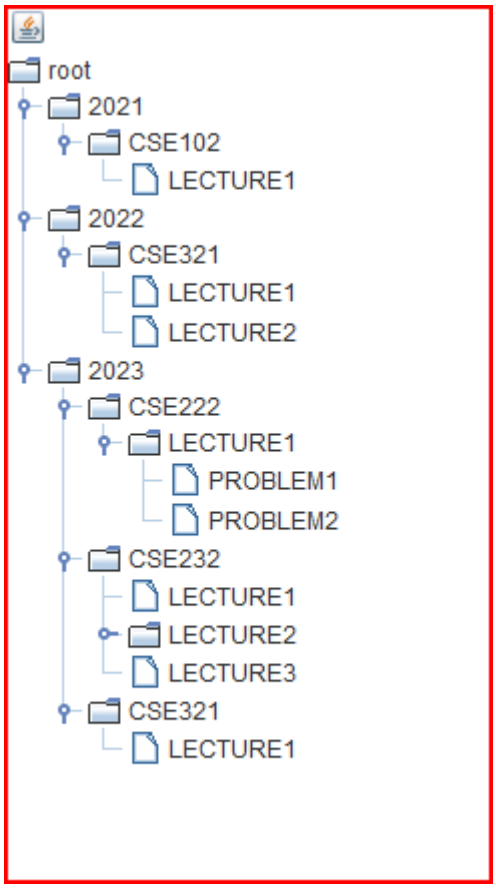
2023->CSE222 to 2024

2023->CSE102->LECTURE 2019

2021->CSE102->LECTURE1 to 2023

6. RUNNING AND RESULTS

Create tree :



BFS Search:

```
Using BFS to find 'CSE224' in the tree ...
height-> 4
Step 1 -> root
Step 2 -> 2021
Step 3 -> 2022
Step 4 -> 2023
Step 5 -> CSE102
Step 6 -> CSE321
Step 7 -> CSE222
Step 8 -> CSE232
Step 9 -> CSE321
Step 10 -> LECTURE1
Step 11 -> LECTURE1
Step 12 -> LECTURE2
Step 13 -> LECTURE1
Step 14 -> LECTURE1
Step 15 -> LECTURE2
Step 16 -> LECTURE3
Step 17 -> LECTURE1
Step 18 -> PROBLEM1
Step 19 -> PROBLEM2
Step 20 -> PROBLEM1
Step 21 -> PROBLEM2
Not found!
```

```
Using BFS to find 'CSE232' in the tree ...
height-> 4
Step 1 -> root
Step 2 -> 2021
Step 3 -> 2022
Step 4 -> 2023
Step 5 -> CSE102
Step 6 -> CSE321
Step 8 -> CSE232(Found!)
```

DFS Search:

```
ENTER A NODE FOR DFS
CSE232
Using BFS to find 'CSE232' in the tree ...
Step 1 -> root
Step 2 -> 2023
Step 3 -> CSE321
Step 4 -> LECTURE1
Step 5 -> CSE232(Found!)
```

```
ENTER A NODE FOR DFS
CSE212
Using BFS to find 'CSE212' in the tree ...
Step 1 -> root
Step 2 -> 2023
Step 3 -> CSE321
Step 4 -> LECTURE1
Step 5 -> CSE232
Step 6 -> LECTURE3
Step 7 -> LECTURE2
Step 8 -> PROBLEM2
Step 9 -> PROBLEM1
Step 10 -> LECTURE1
Step 11 -> CSE222
Step 12 -> LECTURE1
Step 13 -> PROBLEM2
Step 14 -> PROBLEM1
Step 15 -> 2022
Step 16 -> CSE321
Step 17 -> LECTURE2
Step 18 -> LECTURE1
Step 19 -> 2021
Step 20 -> CSE102
Step 21 -> LECTURE1
Not found!
```

Post-Order Search:

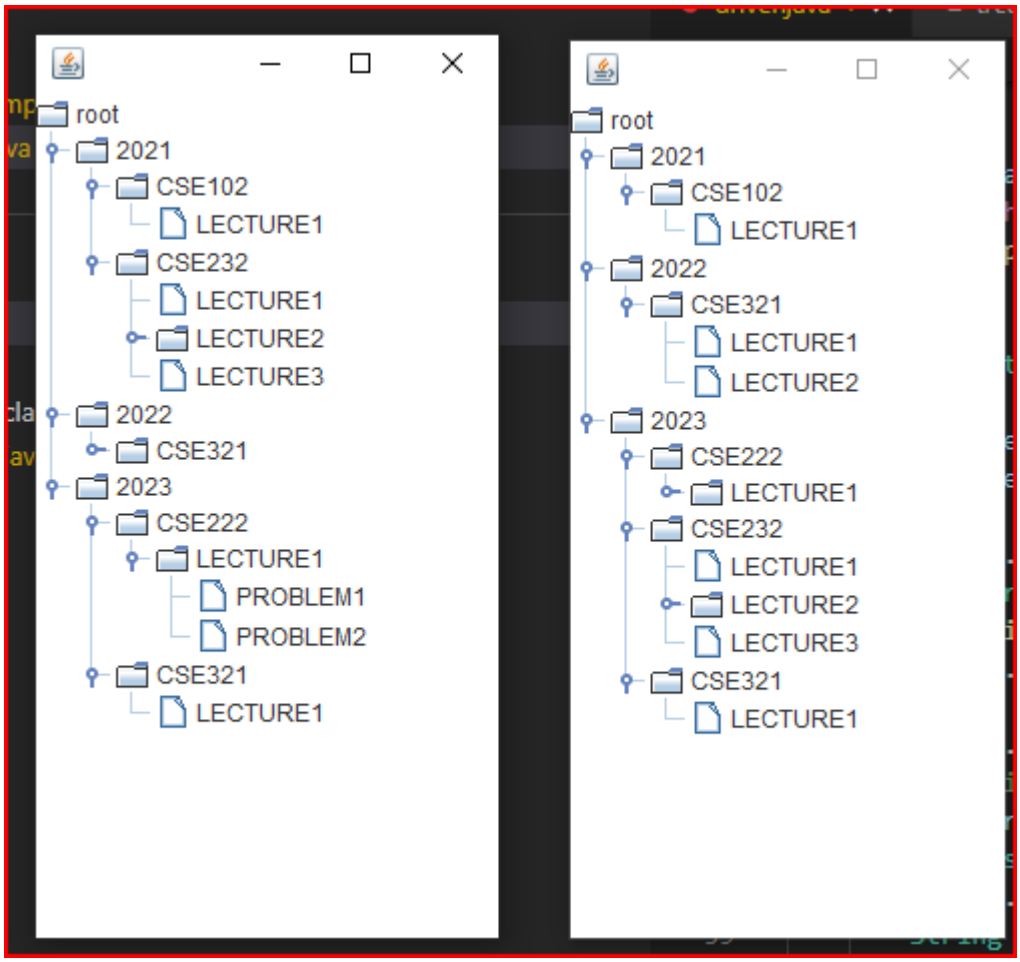
```
ENTER A NODE FOR Post-Order Search
CSE211
Using Post-Order traversal to find 'CSE232' in the tree ...
Step 1 -> LECTURE1
Step 2 -> CSE102
Step 3 -> 2021
Step 4 -> LECTURE1
Step 5 -> LECTURE2
Step 6 -> CSE321
Step 7 -> 2022
Step 8 -> PROBLEM1
Step 9 -> PROBLEM2
Step 10 -> LECTURE1
Step 11 -> CSE222
Step 12 -> LECTURE1
Step 13 -> PROBLEM1
Step 14 -> PROBLEM2
Step 15 -> LECTURE2
Step 16 -> LECTURE3
Step 17 -> CSE232
Step 18 -> LECTURE1
Step 19 -> CSE321
Step 20 -> 2023
Not found!
```

```
ENTER A NODE FOR Post-Order Search
CSE232
Using Post-Order traversal to find 'CSE232' in the tree ...
Step 1 -> LECTURE1
Step 2 -> CSE102
Step 3 -> 2021
Step 4 -> LECTURE1
Step 5 -> LECTURE2
Step 6 -> CSE321
Step 7 -> 2022
Step 8 -> PROBLEM1
Step 9 -> PROBLEM2
Step 10 -> LECTURE1
Step 11 -> CSE222
Step 12 -> LECTURE1
Step 13 -> PROBLEM1
Step 14 -> PROBLEM2
Step 15 -> LECTURE2
Step 16 -> LECTURE3
Step 17 -> CSE232(Found!)
```

1st move:

2023->CSE232 to 2021

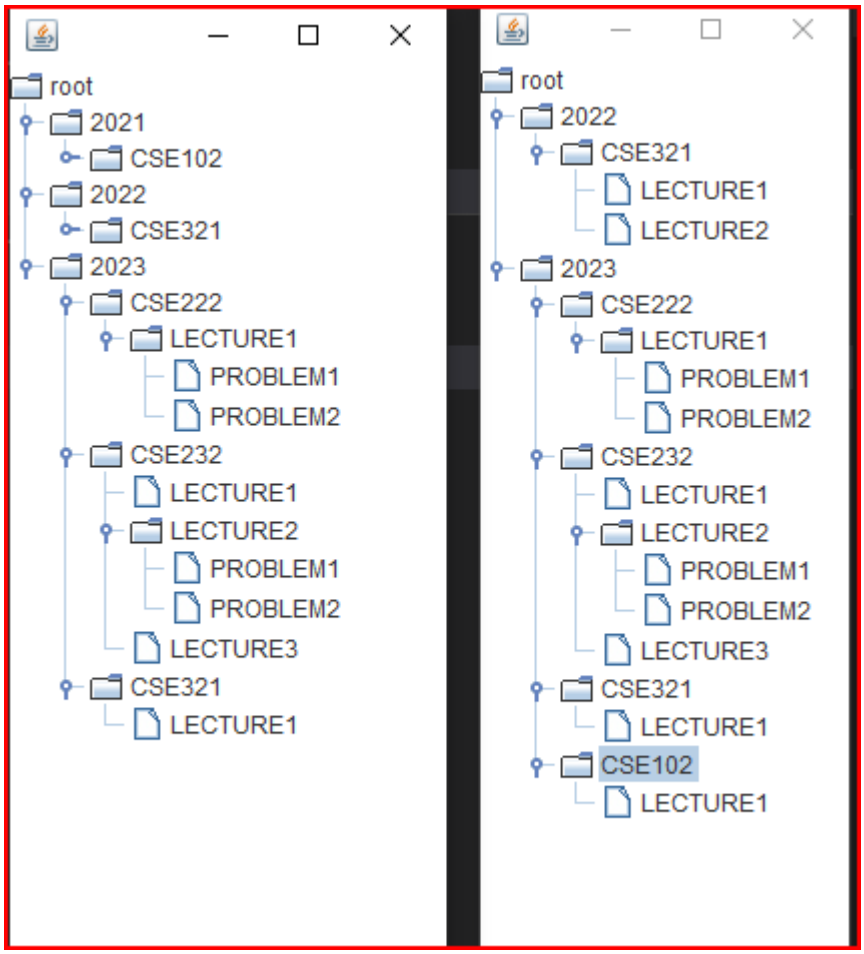
->After moving **->Before moving**



2nd move:

2021->CSE102->LECTURE1 to 2023

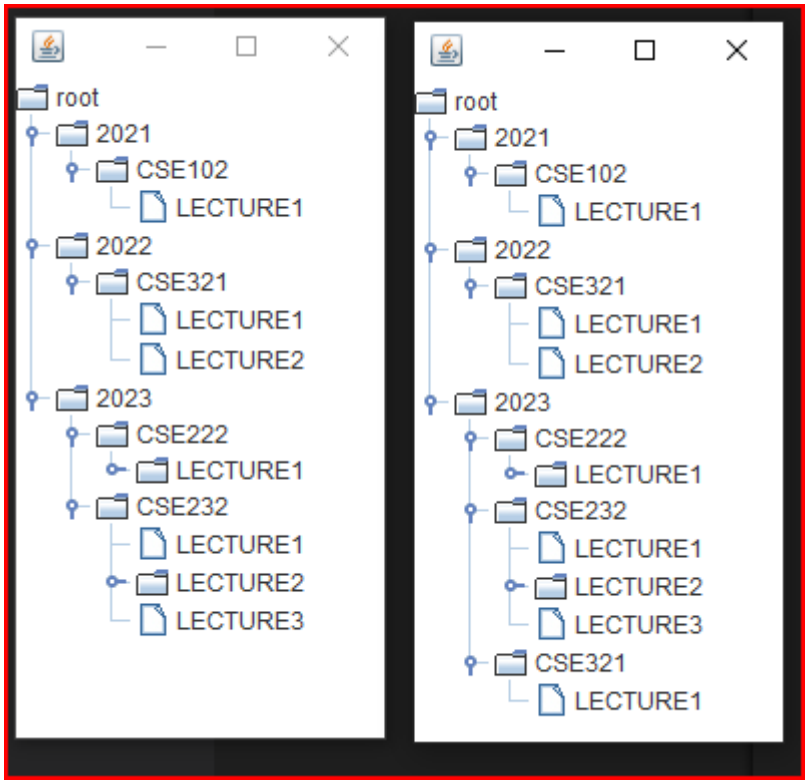
->**before moving** ->**After moving**



3rd move

2023->CSE321->LECTURE1 to 2022

->After move **->Before move**



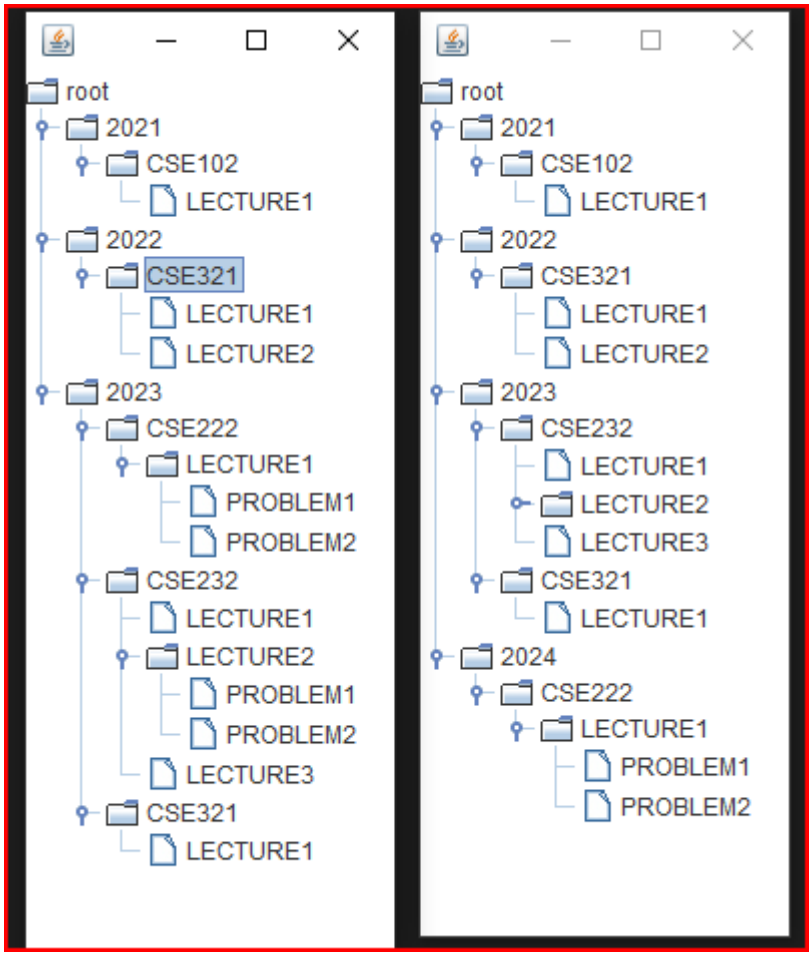
Terminal OUTPUT:

```
ENTER MOVE PATH (  2023;CSE232  !!sample input.input format must this format)
2023;CSE321;LECTURE1
ENTER YEAR TO MOVE
2022
Moved 2023->CSE321->LECTURE1 2022
2023->CSE321->LECTURE1 has ben overwritten
```

4th move

2023->CSE222 to 2024

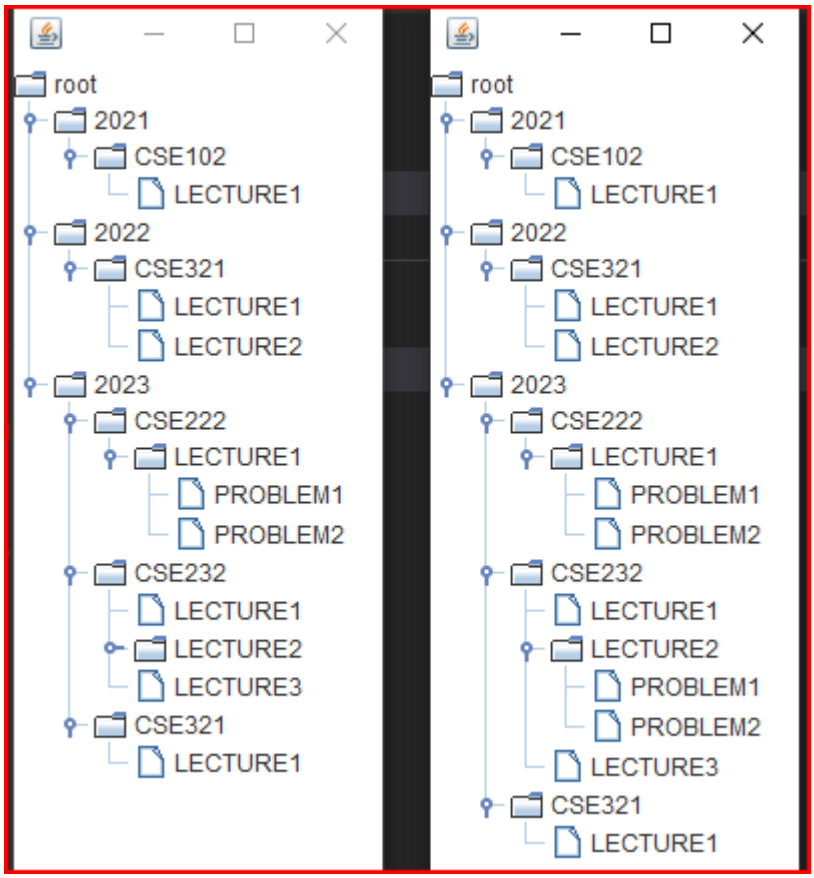
->**Before move** ->**after move**



5th move

2023->CSE102->LECTURE1 2019

->Before move *->After move*



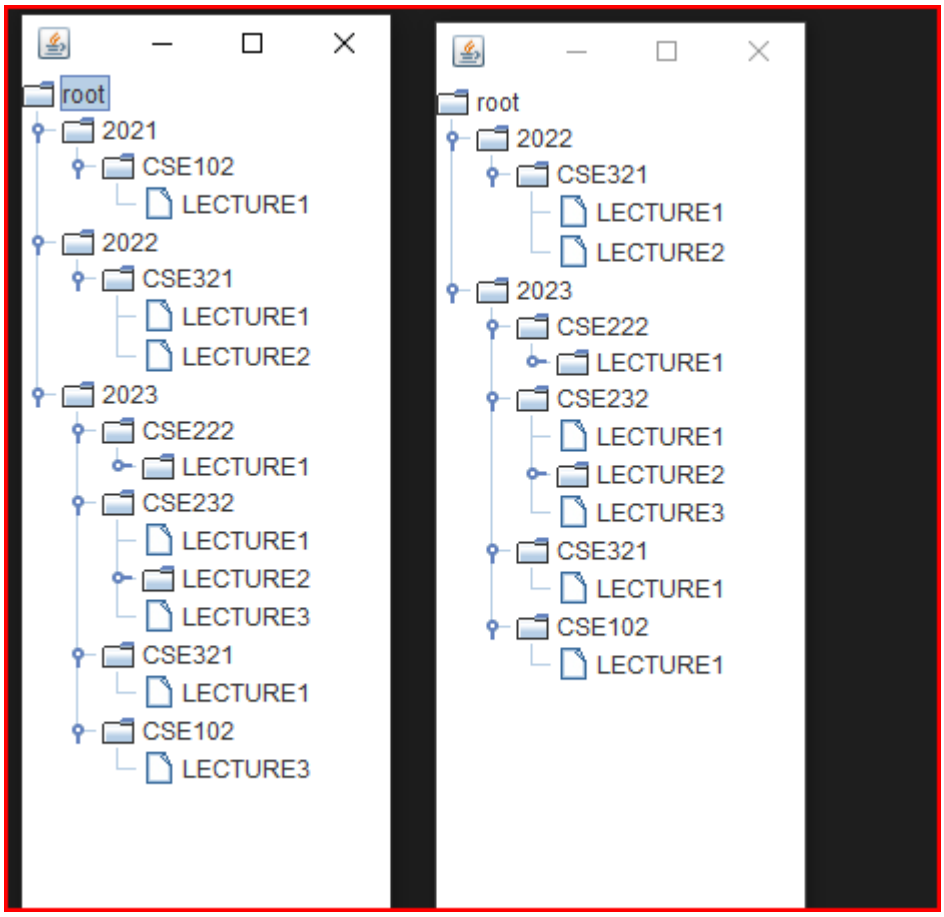
->Terminal OUTPUT

```
ENTER MOVE PATH (  2023;CSE232  !!sample input.input format must this format)
2023;CSE102;LECTURE1
ENTER YEAR TO MOVE
2022
Cannot move 2023->CSE102->LECTURE1 to 2022
it doesn't exist in tree.
```

6th move

2021->CSE102->LECTURE1 to 2023

->Before move *->After move*



Node 2021 and node 2023 has 102 child.but node 102 contain different children . Because of that If user want to move 2021->CSE102->LECTURE1 to node 2023 we must replace them like file operation in real life