GTU Department of Computer Engineering
CSE 222 - Spring 2023
Homework 5 Report

BEYZA ACAR

200104004065

04.05.2023

1-System Requirements

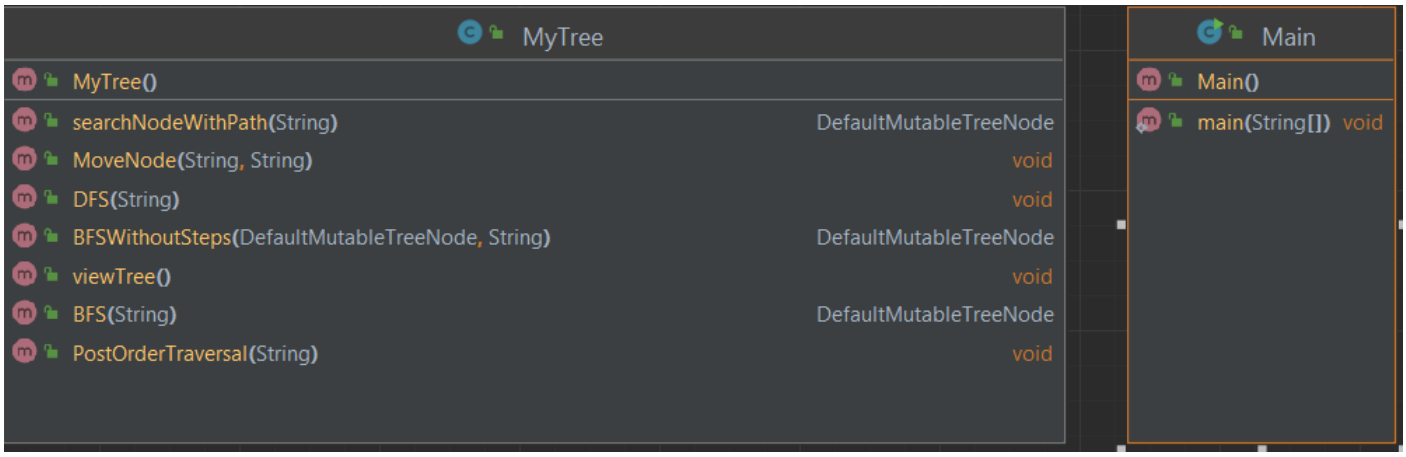
- In this homework;
- PART A:
- We read some information from a .txt file and represent it as a tree structure. In the .txt file, each row represents a data point, and it is split into categories by ";" character.
- We assume that the .txt file has a grid structure, and each column might have an arbitrary number of unique values.
- We parse this information into a tree (not particularly a binary tree) and then print it.
- PART B:
- Apply BFS algorithm to find an input in the tree without using any other external libraries. Print whether the input is found or not, along with the sequence of steps. The input should be given by the user, which is a String.
- PART C:
- Repeat part B with DFS.
- PART D:
- Repeat part B with a post order traversal algorithm.
- PART E:
- Move a node to another location. After the moving operation, we print the tree again.

Important Notes and Assumptions:

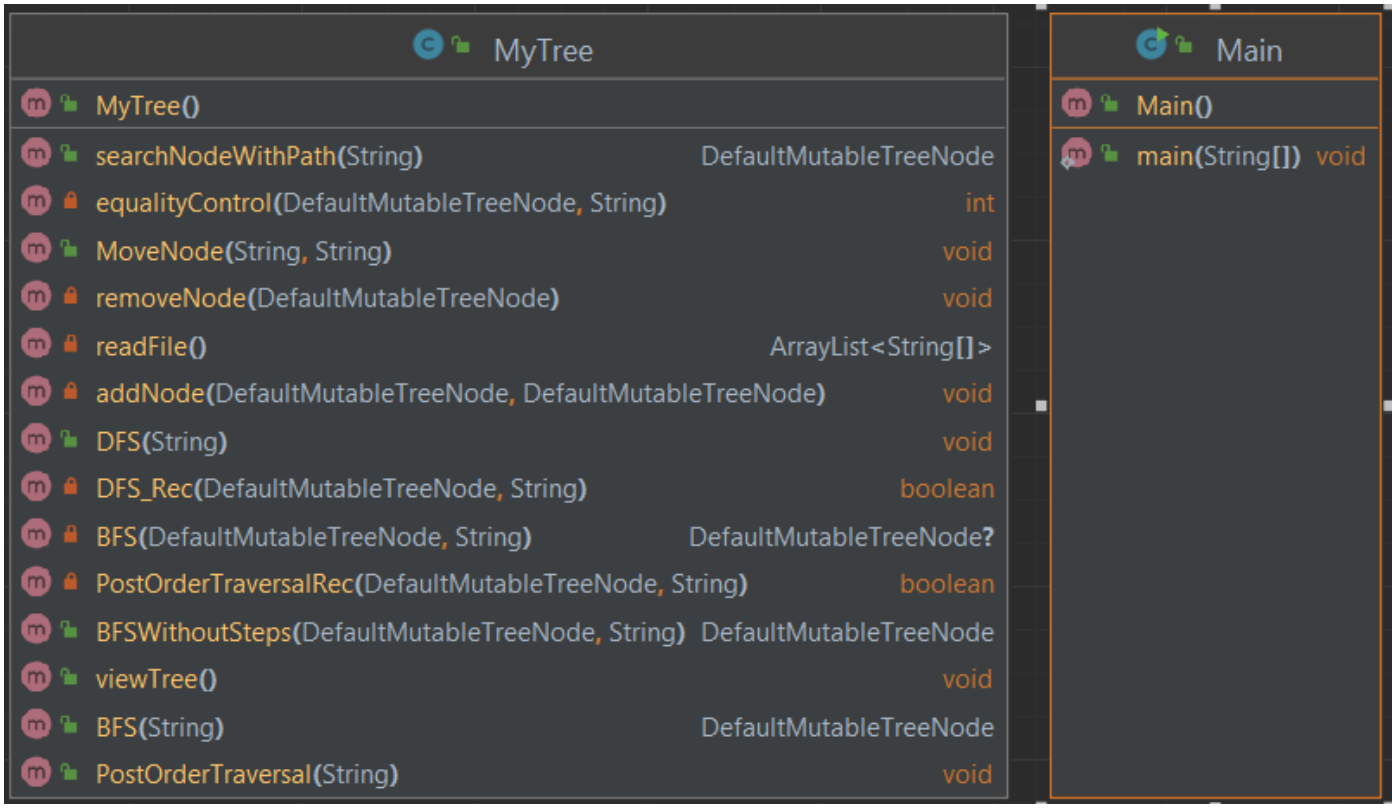
- We assume that each data line has at least 1 column.
- We assume that txt file contains capital letters, numbers and “;" only and the data lines are proper, i.e. there is always a “;” in between two columns.
- We use any auxiliary array to keep track of the data points which were added to the tree. You should work on the tree itself to check it.
- The .txt file should be named as “tree.txt” and it should be in the same folder as the .java files. The file path must be dynamic, i.e. it should run on any computer without any change.

2- Uml Diagram

a. Only public methods



b.All methods



3-Problem Solution Approach

- For PART A : The class constructor calls a method called readFile, the readFile method returns an arrayList of string[]. Then all elements from this 2D String array are placed in the tree (with 2 for loops). During this placement, with the method called equalityControl, before adding it, it is checked whether there is a node with the same value as the node we are about to add among the child nodes, if not, the node is successfully added to the tree. After that a JFrame object is created, and JTree objects is added to this object.
- FOR PART B : I use a queue data structure. Algorithm is like :
    - At first, root is added to the queue
    - And while queue is not empty, move the last element from the queue and print it, think it as "traversed".The elements in the queue represent nodes as "will be traversed".
    - Verbally, my approach is : queue gives us the ability to delete the element we are traversing, so it becomes very easy to navigate the elements without losing their order.
- FOR PART C : I use a recursive algorith fot that part.
    - First step : if root is null do not traverse just return
    - Second step : Visit the current node to check if it is equal with target element
    - Third step : With a for loop, call this method with the rightmost element to the leftmost element in order.
- FOR PART D : this part is similar to PART C
    - First step : if root is null do not traverse just return
    - Second step : With a for loop, call this method with the leftmost element to the rightmost element in order.
    - Third step : Visit current node to check if it is equal with the target element.
- FOR PART E : I have divided the path into two parts: the upper and lower parts.
    - In the addNode method, starting from the node to be added, I recorded the steps I took towards the root by using a stack. I did

not touch the remaining part since it would be added as a whole. I used a stack structure because the path needed to be traversed from bottom to top, while my path was from top to bottom.

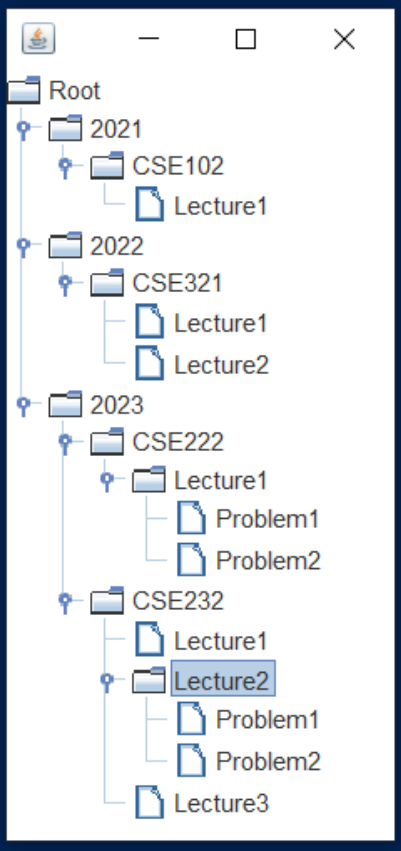
- In the remove method, I checked whether the node being deleted had only one child while deleting it. If it had only one child, I made the parent node the current node and repeated these checks.

4- Test Cases – Running and Results

a. SEACH ALGORITHMS (PART A,B,C,D)

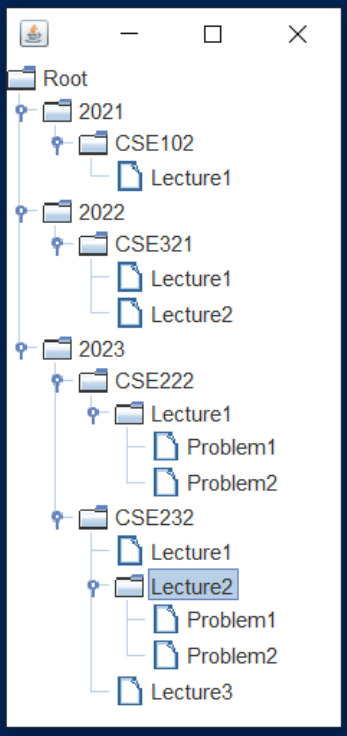
input : CSE232

```
Enter the string you want to add:
CSE232
--------------BFS ALGORITHM--------------
Step 1: Root
Step 2: 2021
Step 3: 2022
Step 4: 2023
Step 5: CSE102
Step 6: CSE321
Step 7: CSE222
Step 8: CSE232
Found
--------------DFS ALGORITHM--------------
Step 1: Root
Step 2: 2023
Step 3: CSE232
Found
--------------POSTORDER TRAVERSAL--------------
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
```



input = k

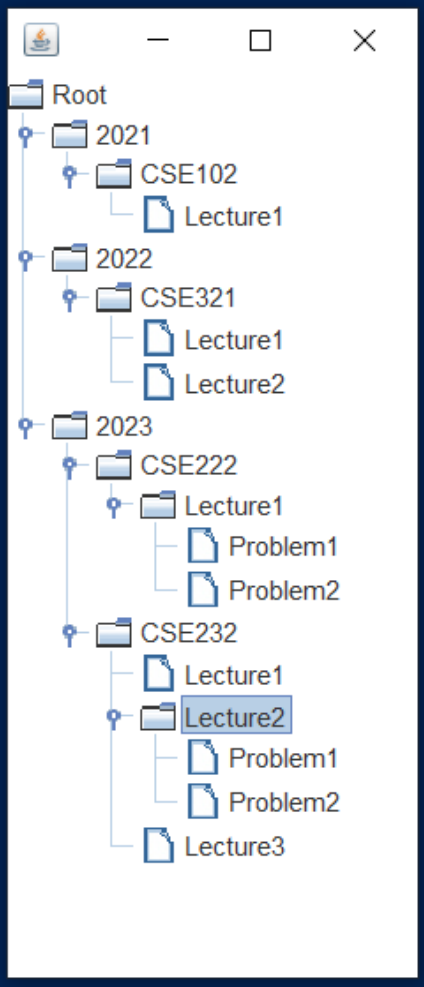
```
Enter the string you want to add:
k
--------------BFS ALGORITHM--------------
Step 1: Root
Step 2: 2021
Step 3: 2022
Step 4: 2023
Step 5: CSE102
Step 6: CSE321
Step 7: CSE222
Step 8: CSE232
Step 9: Lecture1
Step 10: Lecture1
Step 11: Lecture2
Step 12: Lecture1
Step 13: Lecture1
Step 14: Lecture2
Step 15: Lecture3
Step 16: Problem1
Step 17: Problem2
Step 18: Problem1
Step 19: Problem2
Not found
--------------DFS ALGORITHM--------------
Step 1: Root
Step 2: 2023
Step 3: CSE232
Step 4: Lecture3
Step 5: Lecture2
Step 6: Problem2
Step 7: Problem1
Step 8: Lecture1
Step 9: CSE222
Step 10: Lecture1
Step 11: Problem2
Step 12: Problem1
Step 13: 2022
Step 14: CSE321
Step 15: Lecture2
Step 16: Lecture1
Step 17: 2021
Step 18: CSE102
Step 19: Lecture1
Not found
```




```
--------------POSTORDER TRAVERSAL--------------
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: 2023
Step 19: Root
Not found
```

Input : CSE102

```
Enter the string you want to add:
CSE102
--------------BFS ALGORITHM--------------
Step 1: Root
Step 2: 2021
Step 3: 2022
Step 4: 2023
Step 5: CSE102
Found
--------------DFS ALGORITHM--------------
Step 1: Root
Step 2: 2023
Step 3: CSE232
Step 4: Lecture3
Step 5: Lecture2
Step 6: Problem2
Step 7: Problem1
Step 8: Lecture1
Step 9: CSE222
Step 10: Lecture1
Step 11: Problem2
Step 12: Problem1
Step 13: 2022
Step 14: CSE321
Step 15: Lecture2
Step 16: Lecture1
Step 17: 2021
Step 18: CSE102
Found
--------------POSTORDER TRAVERSAL--------------
Step 1: Lecture1
Step 2: CSE102
Found
```



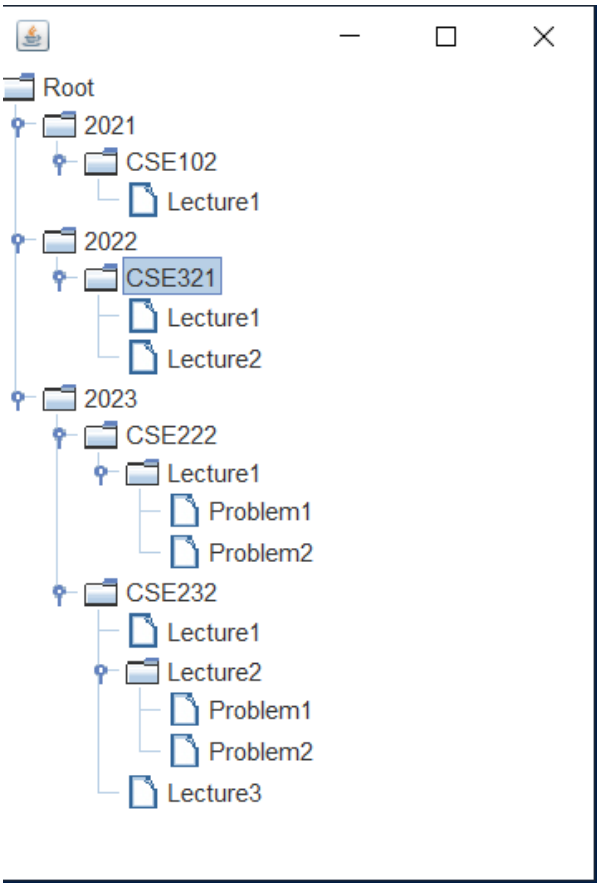
b. ***Move Method (PART E)***

First input : 2023,CSE222,Lecture1

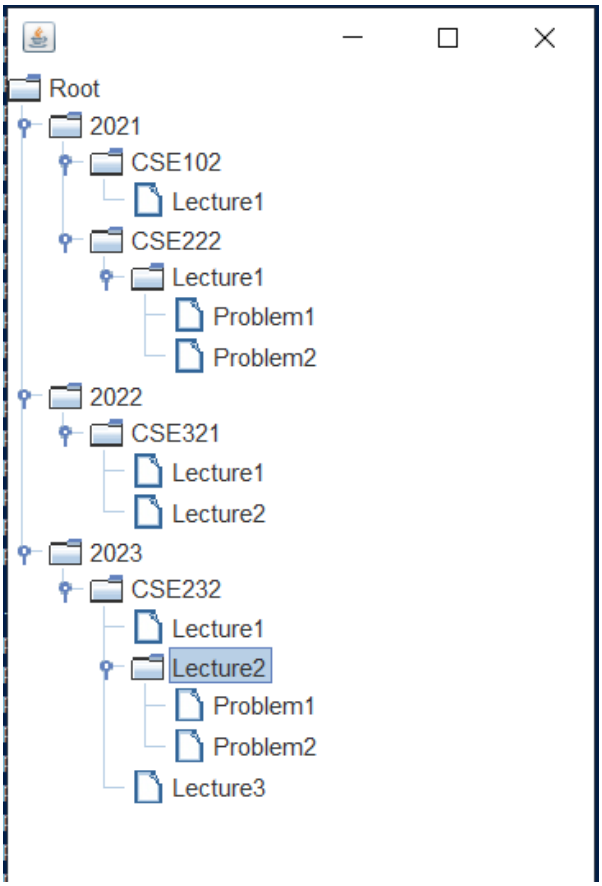
Second input : 2021

```
Enter the path you want to move to another path :
2023,CSE222,Lecture1
Enter the path you want to place the node you have moved.
2021
```

THE ORIGINAL TREE



THE EDITED TREE

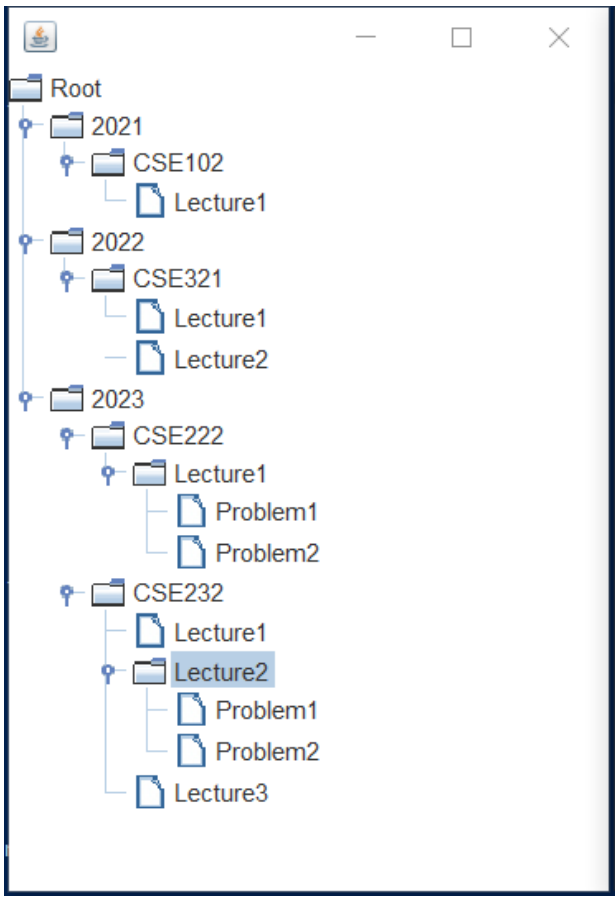


First input : 2022,CSE321,Lecture2

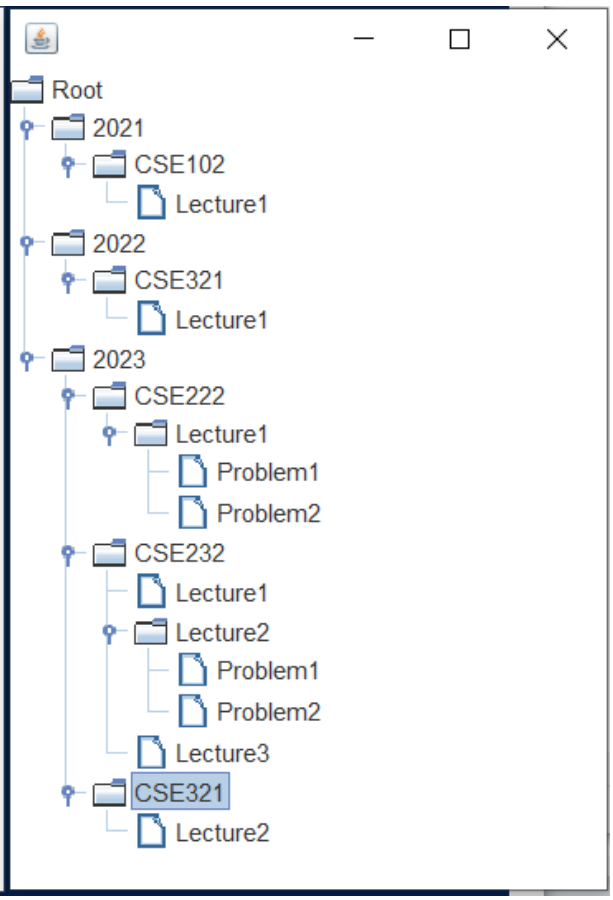
Second input : 2023

```
----------------MOVE----------------
Enter the path you want to move to another path :
2022,CSE321,Lecture2
Enter the path you want to place the node you have moved.
2023
```

THE ORIGINAL TREE



THE EDITED TREE

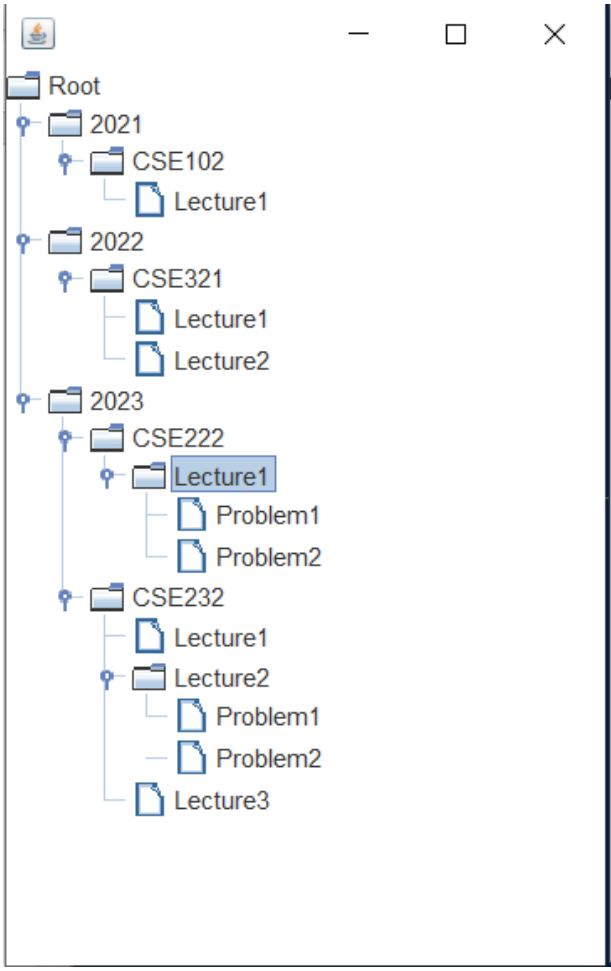


First input : 2023,CSE232,Lecture2,Problem2

Second input : 2022

```
----------------MOVE----------------
Enter the path you want to move to another path :
2023,CSE232,Lecture2,Problem2
Enter the path you want to place the node you have moved.
2022
```

THE ORIGINAL TREE



THE EDITED TREE