National University of Computer and Emerging Sciences, Lahore Campus 0542

	Course Name:	Data Structure Lab	Course Code:	CL2001
	Program:	CS(All Sections), DS(All Sections), BSSE-3AI	Semester:	Fall 2023
	Duration:	2 hours 45 minutes	Total Marks:	100
	Exam Date:	05-01-2024	Weight:	45 %
	Section:		Page(s):	5
	Exam Type:	Lab Final		

Student Name: Tayyab Sami Roll No. 22i-2505 Section: 3A-SE

Instruction/Notes:

- We will check your code for plagiarism. If plagiarism is found, it will result in F grade in lab.
- No cell phones are allowed. Sharing of USBs or any other items is not allowed.
- You are not allowed to have any helping code with you.
- Submission Path is \\Cactus1\Xeon\Fall 2023\DS Lab Final Exam Submission \ "your section "
- Submit Paper in respective sections only. No query will be entertained later
- Understanding the questions is also part of it. You can make any useful assumptions, where required. Provided main functions can help.

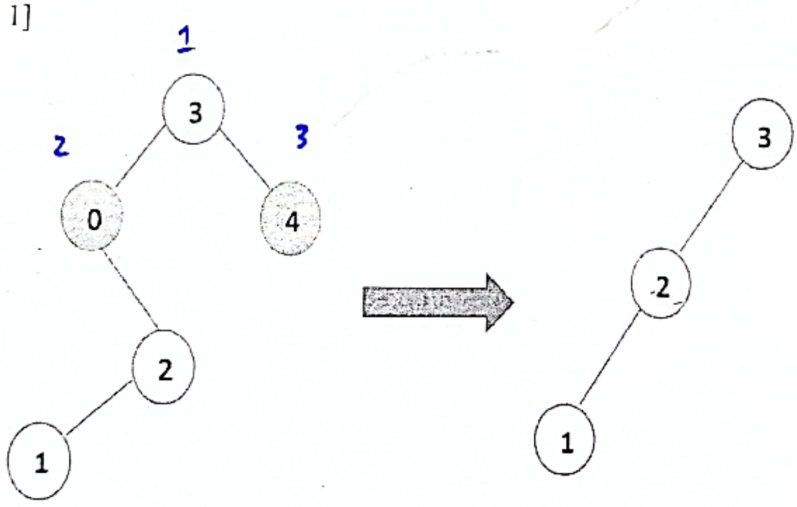
Question 1: [Marks - 30]

Given a binary search tree, write a function that takes root, low and high values as input and trim the tree so that all its elements lie in [low-high] range and then return the trimmed BST. Trimming the tree should not change the relative structure of the elements that will remain in the tree (i.e., any node's descendant should remain a descendant).

Example 1:

Input: root = [3, 0, 2, 1, 4], low = 1, high = 3

Output: [3 , 2, null, 1]



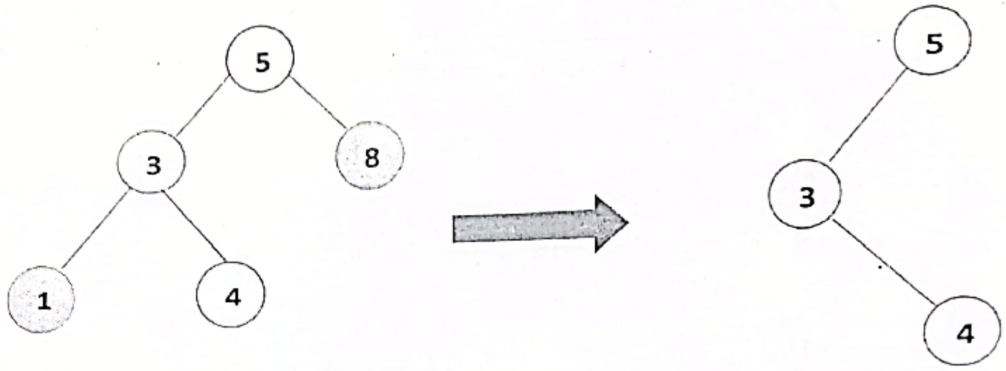
1 of 5

Example 2:

Input: root = [5, 8, 3, 4, 1], low = 3, high = 5

Output: [5, 3, null, null, 4]

5 3



5

Question 2: [Marks - 30] 0542

In the city population management system, you are tasked with finding the smallest range that encompasses population changes across the top K cities. Each row in the 2D hash map represents a city, identified by its name. Within each city, there is a list that contains the population data for different years.

Your goal is to implement an algorithm that efficiently identifies the smallest range of population changes across the first K cities in the hash map.

1. **Initialization:**
   The 2D hash map is initialized with a fixed number of rows to represent cities.
2. **Insert Functionality:**
   The system receives population data for a specific city for a given year. If this is the first population data for the city, use quadratic probing to find the next available row. Once the city's row is determined, use linear probing to find the next available slot within the row for the population data.
3. **Delete Functionality:**
   Population data for a certain year in a city needs to be updated or corrected. Implement a function to delete the existing population data from the specified row and year. Use linear probing to search for the data within the row.
4. **Search Functionality:**
   Authorities want to check the population of a city for a particular year. Implement a function to search for the population data in the specified row and year. Use linear probing within the row to find the data. If the data is not found, return a message indicating that the data is unavailable.
5. **Find Smallest Range Functionality:**
   Implement an algorithm to find the smallest range containing elements from the first K cities population.
   Example:
   K=3
   City1:{4,7,9,12,15}
   City2:{0,8,10,14,20}.
   City3:{6,12,16,30,50}.
   Output: The smallest range is [6 8].
   Explanation: The smallest range is formed by the number 7 from the first city, 8 from the second city, and 6 from the third city.
6. **Show Hash Map:**
   Implement a function to display the current state of the 2D hash map. This should include the names of the cities (row keys) and the corresponding lists of population data for different years.

Details:

- The hash map is initially configured to accommodate around 10 cities, and each city's population data is tracked for max 7 years.
- The first population data insertion for a city involves quadratic probing to find the row and linear probing to find the slot within the row.

city1 77
city2 84
city3 91
98

Question 3: [Marks - 4

Consider, a digital representation of a traditional Pakistani family tree where each node represents a fa member. This time, the tree is structured based on the lifetime earnings of each family member, with eac node containing information about the person's name, age, lifetime earnings, and role in the family.

1. **Insert Functionality:**

Implement a recursive insert function to add a new family member to the hierarchy. Each family member has a unique identifier, a name, age, lifetime earnings, and a designated role (e.g., father, mother, son, daughter). Ensure that the new member is inserted in a way that maintains the family hierarchy based on lifetime earnings.

2. **Delete Functionality:**

Implement a recursive delete function to remove a family member from the hierarchy based on their unique identifier.

3. **Search Functionality:**

Implement a recursive search function to find information about a family member based on their unique identifier.

4. **PromoteToRoot Functionality:**

Implement a function named **PromoteToRoot** that takes a family member's unique identifier and forcefully promotes the corresponding node to the root of the tree. This promotion is based on the family member's extraordinary efforts and achievements. The tree should be reorganized to maintain the family hierarchy based on lifetime earnings. During the promotion, ensure that the Binary Search Tree (BST) properties are preserved.

5. **BurnNode Functionality:**

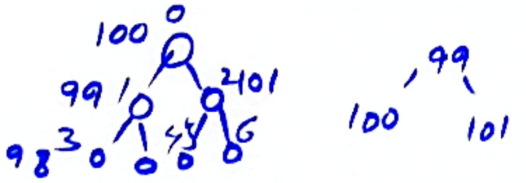
Implement a function named **BurnNode** that takes a family member's unique identifier and removes the corresponding node and its descendants from the tree. Ensure that the removal process is handled properly to maintain the family tree structure based on lifetime earnings.

6. **Sibling Level Sum Difference:**

Write a function named LevelSumsDifference that calculates and prints the absolute difference between the sums of lifetime earnings for nodes at odd levels and nodes at even levels in the family tree.

7. **Boundary Traversal:**

Write a function named BoundaryTraversal that prints the boundary nodes of the family tree based on lifetime earnings. The boundary nodes include the leftmost nodes of each level, the leaf nodes, and the rightmost nodes of each level, excluding duplicates.

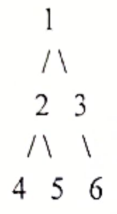


Connect Siblings:

Implement a function named **ConnectSiblings** that connects nodes of each level in the family. (For this task you have to add the next property in the node)

For Example:

Input:



Output: