National University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures	Course Code:	CS2001
Program:	BS (CS, SE, DS)	Semester:	Fall 2021
Duration:	60 Minutes	Total Marks:	20
Paper Date:	03-Dec-2021	Page(s):	6
Sections:	ALL	Section:	[redacted]
Exam:	Sessional 2	Roll No:	[redacted]

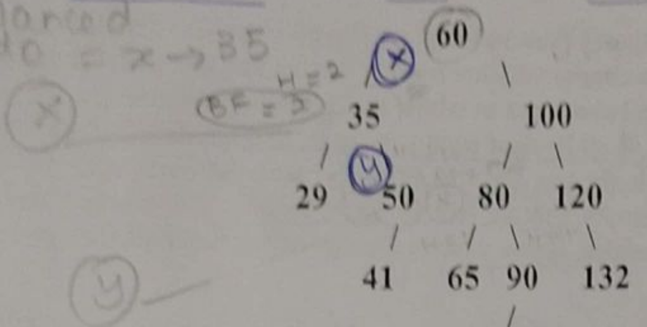
11

Instruction/Notes: Answer in the space provided. You cannot ask for rough sheets since they are already attached. Rough sheets **will not be graded or marked.** In case of any confusion or ambiguity, make a reasonable assumption.

Question 1: **(Marks:5+5)**

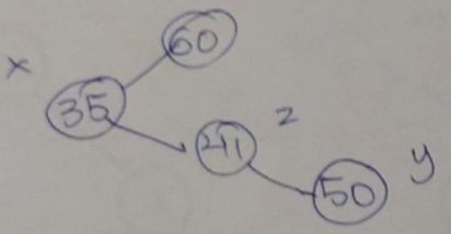
a) Redraw the following AVL tree after deletion of key 29 and. You must show all working including the names of disbalanced cases, nodes, and the rotations performed.

~~imbalanced node = x → 35~~

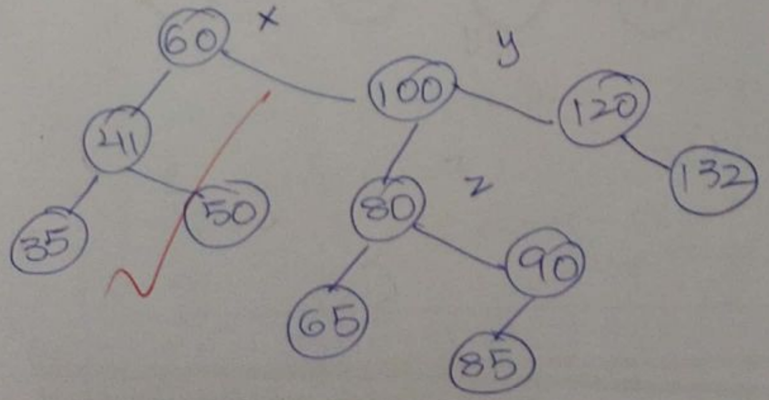


imbalancenode = 35

1 case: double right left rotation
(left-right imbalance)
First rotate y, z to right

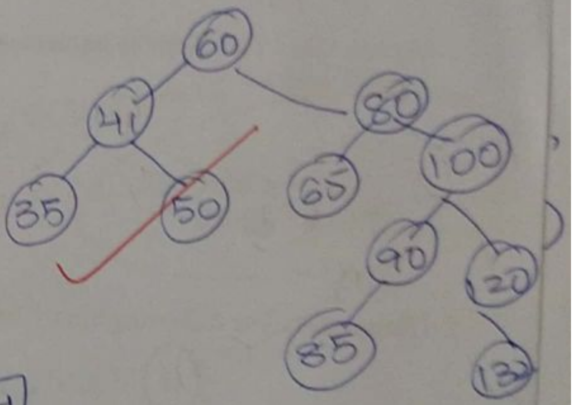


2 then rotate x, z left

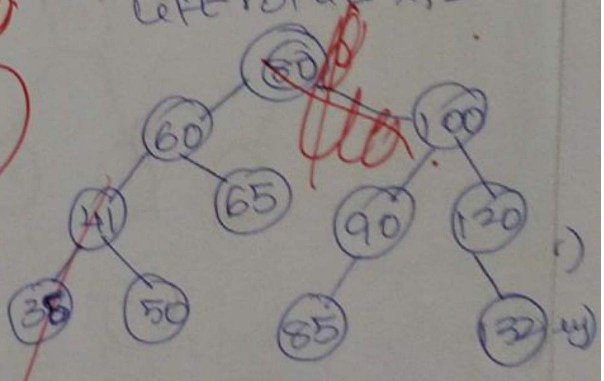


3 imbalanced node = 60
BF = 2
case: double right left
(left-right imbalance)

First right rotate y, z



4 second left rotate x, z



now all balanced

Scanned with CamScanner

Roll Number: 20L-1080

Section: ______

b) For each of the scenarios given below, suggest the most appropriate data structure chosen from the (Arrays, doubly linked-list, Queue, Stack, tree)

1. to record the sequence of all the pages browsed in one session.	stack
2. to store information about the directories and files in a system	~~Linklist~~ doubly linked list
3. to implement printer spooler so that jobs can be printed in the order of their arrival.	queue ✓
4. to implement "back" functionality in the internet browser.	~~sta~~ stack ✓
5. to store an image in the form of a bit map	tree

(Marks: 10)

Question 2:

Write a recursive C/C++ function **isDifferenceBinaryTree** in the class BinaryTree. This function is passed the root of the binary tree as a parameter. It then checks whether a given binary tree is a **difference binary tree** or not. We define difference binary tree as a binary tree in which the difference between the sum of all keys of left subtree of a **non-leaf** node and sum of all keys of right subtree of that non-leaf node equals the key of that node. If every non-leaf node in a binary tree has that property, then the binary tree will be a difference binary tree. The sum of keys of an empty subtree will be equal to zero. If the binary tree is a difference binary tree, then return true else return false. If you want to use any helper function, then you must give its implementation as well.

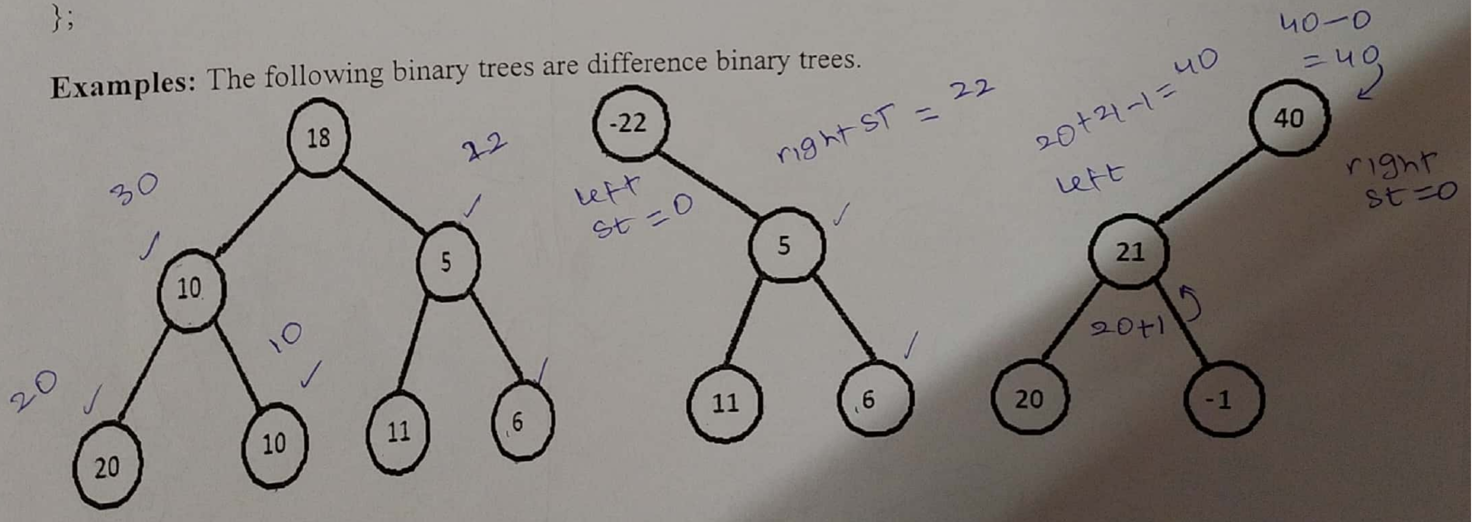
Assume that Tree Node is implemented as follows and BinaryTree is a friend class of TNode:

only
* nonleaf ↓

```
class TNode
{
    int key;
    TNode* lChild;
    TNode* rChild;
};
```

left – right

Examples: The following binary trees are difference binary trees.



30
22
18

30
11
19

① bool is DifferenceBinaryTree();
② bool IS Difference BT (Node*);
③ bool compute Sum (Node*, Node*, Node*);

Roll Number: 20L-1080 Section: BSSE3A

```
bool  is Difference Binary Tree ( )
  {
        if ( is Difference BT ( root) == true)
              return true;

        else
              return False;

  }
```

3+1

```
bool  is Difference BT ( Node* curr) ~~int sum~~
  {
        ~~if (curr == 0)~~
              ~~return true;~~

        if (curr)
           {
              is Difference BT (curr->left);
              if (Compute Sum (curr->left, curr->right,
                                curr) == ~~true~~ False)
                 return false;
              is Difference BT (curr->right);
           }

        return true;
  }
```

Sum of all key just wr left & right child

X.

```
bool  Compute Sum ( Node* currLeft, Node* currRight,
                                     Node* curr)
{
     if (curr)
     {     if (currLeft->key + currRight->key == curr->key)
                return true; else return False;
     }
}  return true;
```

FAST School of Computer Science

Scanned with CamScanner