data structures

assignment 1

Sandesh Shrestha
2K21/CO/417
A6 Batch

Sandesh Shrestha 2K22/CO/427

Topic..... Date.....

2) Write a program to construct a binary search tree on given pre-order traversal of tree. After constructing the Binary search tree, print the nodes level wise top to down and left to right.

→ C++ program

```
#include <iostream>
#include <vector>
#include <queue>
#include <climits>
using namespace std;
struct TreeNode {
        TreeNode *left;
        TreeNode *right;
        int val;
        TreeNode();
        TreeNode(int a) : val(a) { };
};

TreeNode* build(vector<int> &A, int &i, int ub)
{
  if(i==A.size() || A[i] > ub)
        return NULL;
  TreeNode *root = new TreeNode(A[i++]);
        root->left = build(A, i, root->val);
        root->right = build(A, i, ub);
  return root;
}
```

Teacher's Sign .....

Topic..... Date.....

```
// build function
TreeNode* bstfromPreorder(vector<int>&array)
  {
   int i=0;  // first element of array
   return build(@rray, i, INT_MAX);
}

//function for printing levelorder traversal
void levelOrder(TreeNode* root)
{
   if (root==NULL) return;
  queue<TreeNode*> q;
   q.push(root);
while(!q.empty()){
        int size=q.size();
        for(int i=0; i<size; i++){
            TreeNode* node=q.front();
            cout<<node->val<<" ";
            q.pop();
          if(node->left!=NULL){
                  q.push(node->left);}
          if(node->right!=NULL){
                  q.push(node->right);}
        }
        cout<<endl;  //new line for new level
    }
}
int main(){
  vector<int> array={8,5,1,7,10,12,~~13~~};
  TreeNode *root=bstfromPreorder(array);
  cout<<"\n levelorder traversal is:"<<endl;
  levelOrder(root);
  return 0;
}
```

Teacher's Sign .....

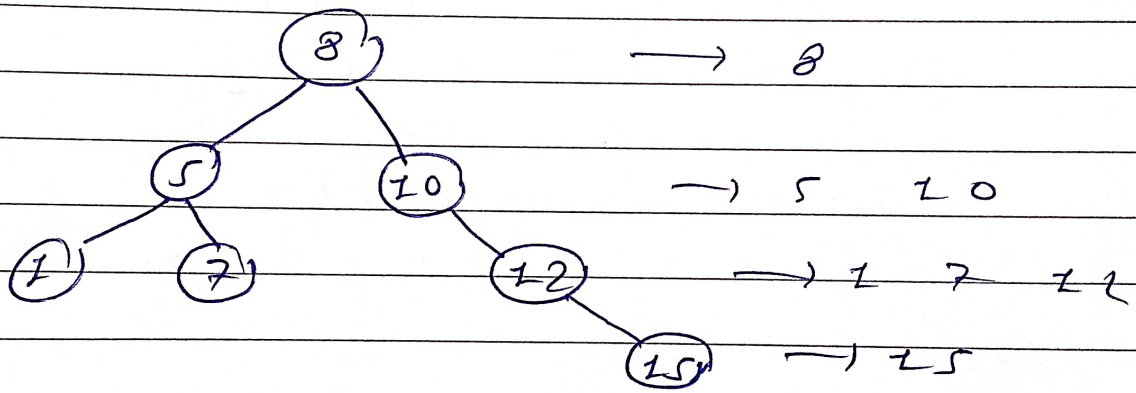
Topic.....Date.....

Output:

The level order traversal is:

8
5 10
1 7 12
15

The actual tree looks like:



```cpp
#include <iostream>
#include <vector>
#include <queue>
#include <climits>
using namespace std;
struct TreeNode
{
    TreeNode *left;
    TreeNode *right;
    int val;
    TreeNode();
    TreeNode(int a) : val(a){};
};

// input array linxa ani pointer for each element of that array ani upper bound linxa
TreeNode *build(vector<int> &A, int &i, int upper_bound)
{
    if (i == A.size() || A[i] > upper_bound)
        return NULL;                          // if array ko sabai element insert gari sakyo vane or insert grne element is bigger than upperbound
    TreeNode *root = new TreeNode(A[i++]);  // create a new node with value A[i]the element
    root->left = build(A, i, root->val);    // for left child upper bound is the root ko value
    root->right = build(A, i, upper_bound); // for right ko child upper bound is root ko upper bound
    return root;
}
TreeNode *bstFromPreorder(vector<int> &Array)
{
    int i = 0; // array ko first element
    return build(Array, i, INT_MAX);
}

void inorder(TreeNode *root)
{
    if (root != NULL)
    {
        inorder(root->left);
        cout << root->val << " ";
        inorder(root->right);
    }
}

/*algo is suru ma queue maintain garne tesma one by one send garne node hru
first ma root push grne queue ma tespxi teslai pop ani print grne then root ko child xa vane teslai push grne ani repeat until the whole queue is empty;*/
void LevelOrderTraversal(TreeNode *root)
{
    if (root == NULL)
        return;
    queue<TreeNode *> q; // maintain a queue
    q.push(root);        // push the root value
    // until whole queue is empty i.e. sabai elements printed
    while (!q.empty())
    {
        int size = q.size(); // harek level ma kati ota element xa is stores in size
        for (int i = 0; i < size; i++)
```

```
54          {                                   // jati ota element xa euta level ma teti ota
ko lagi loop
55              TreeNode *node = q.front(); // front ko node i.e current node
56              cout << node->val << " ";   // print the value of node
57              q.pop();                    // remove it out of stack
58              if (node->left != NULL)
59              {
60                  q.push(node->left);
61              } // push left child if exists
62              if (node->right != NULL)
63              {
64                  q.push(node->right);
65              } // push right child if exists
66
67          }
68          cout << endl; // new line for new level
69
70      }
71  }
72
73  int main()
74  {
75      vector<int> array = {8, 5, 1, 7, 10, 12,15};
76      TreeNode *root = bstFromPreorder(array);
77      cout << "\nThe inorder traversal is: " << endl;
78      inorder(root);
79      cout << "\n\n The level order traversal top to down & left to right  is: " <<
endl;
80      LevelOrderTraversal(root);
81      return 0;
82  }
```

```
PROBLEMS 51  OUTPUT  DEBUG CONSOLE  TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS D:\STUDY MATERIALS\SEM III\DSA\codes> cd "d:\STUDY MATERIALS\SEM III\DSA\codes\C\" ; if ($?) { g++ tree_From_preorder.cpp -o tree_From_preorder }
m_preorder }

The inorder traversal is:
1 5 7 8 10 12 15

 The level order traversal top to down & left to right  is:
8
5 10
1 7 12
15
PS D:\STUDY MATERIALS\SEM III\DSA\codes\C>
```

Topic.....Date.....

data structures assignment - I

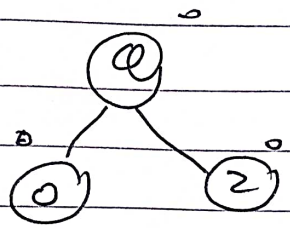
Sandesh Shrestha
Roll: 2K21/CO/417
Batch: A6

Question 1

Insert Q, Z, O, P, W, X, Y, V, S, U, T and R into initially empty AVL tree. From the tree, thus obtained delete W. Show all the intermediate trees/steps and write the appropriate rotations with proper justification /explanation.

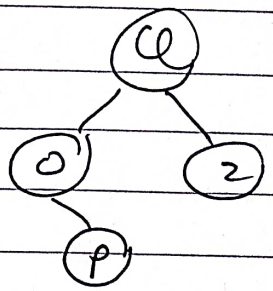
Topic.....Date.....

#) Insert Q and then Z and O

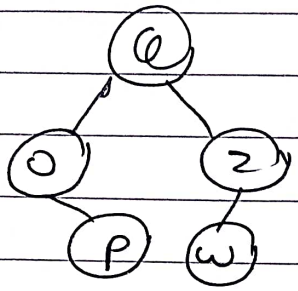


Balanced

(ii) insert P

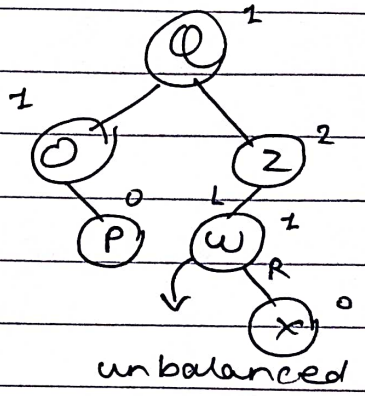


(iii) insert W



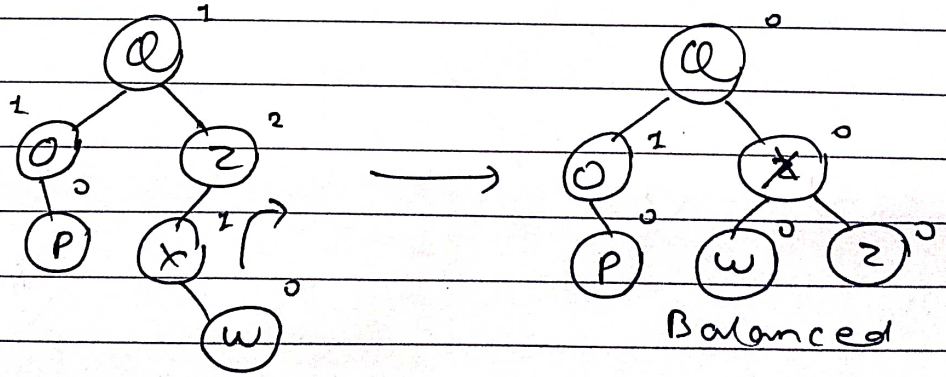
Balanced

(iv) insert X



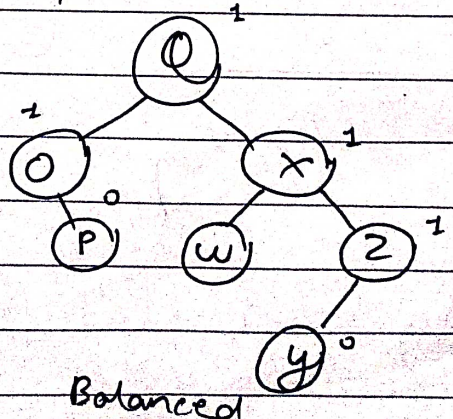
unbalanced

after LR rotation,



Balanced

(v) insert Y

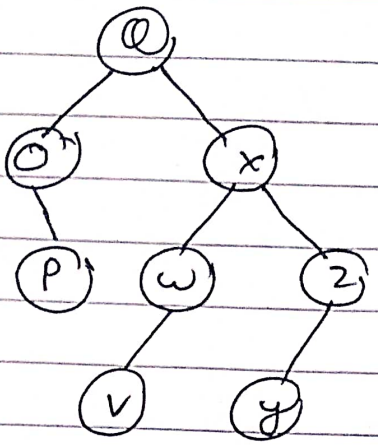


Balanced

Teacher's Sign.....

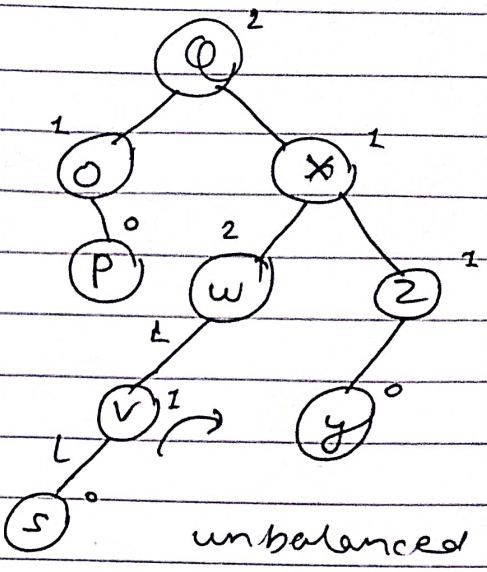
Topic.....Date.....

(vi) insert v



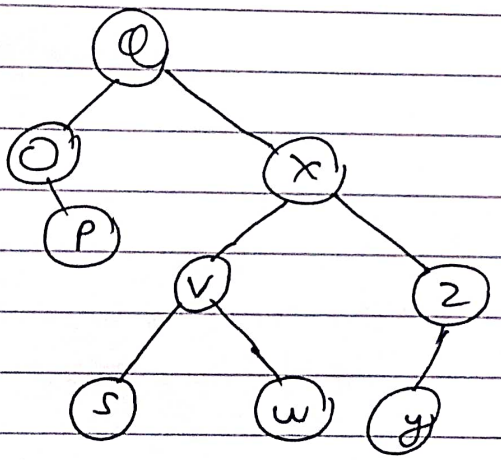
Balanced

(vii) insert S



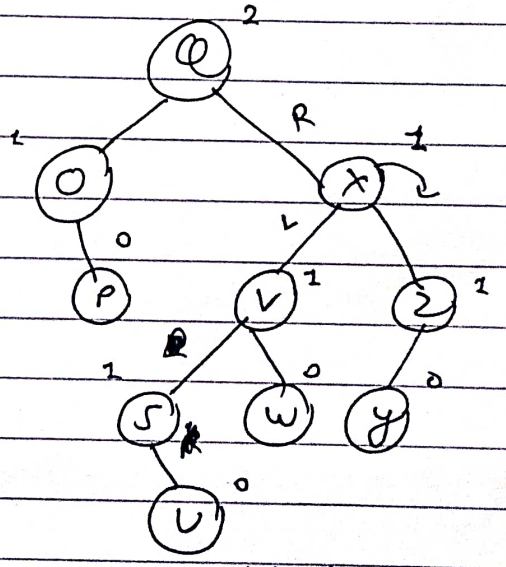
unbalanced

doing LL Rotation



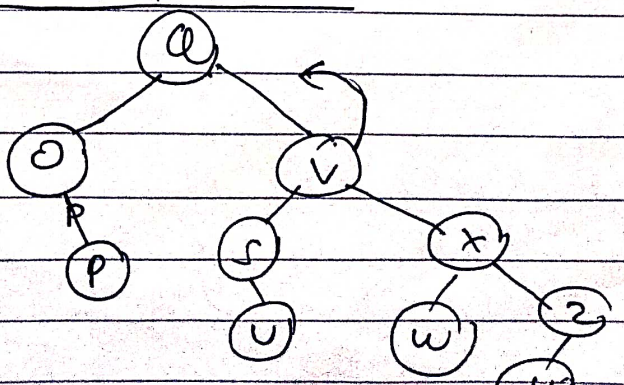
Balanced

(viii) insert U



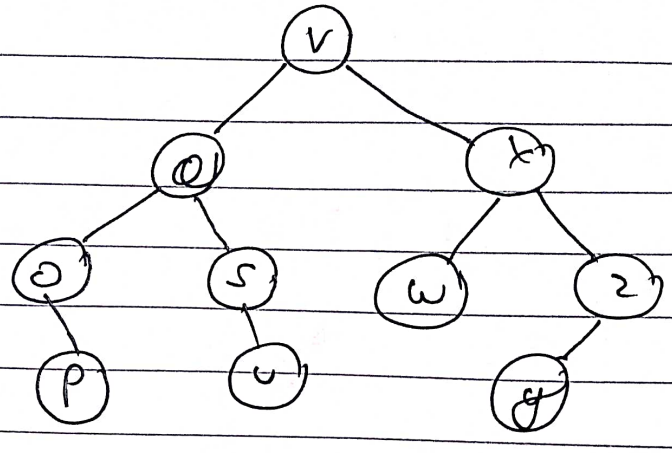
unbalanced

doing RL rotation



Teacher's Sign .....

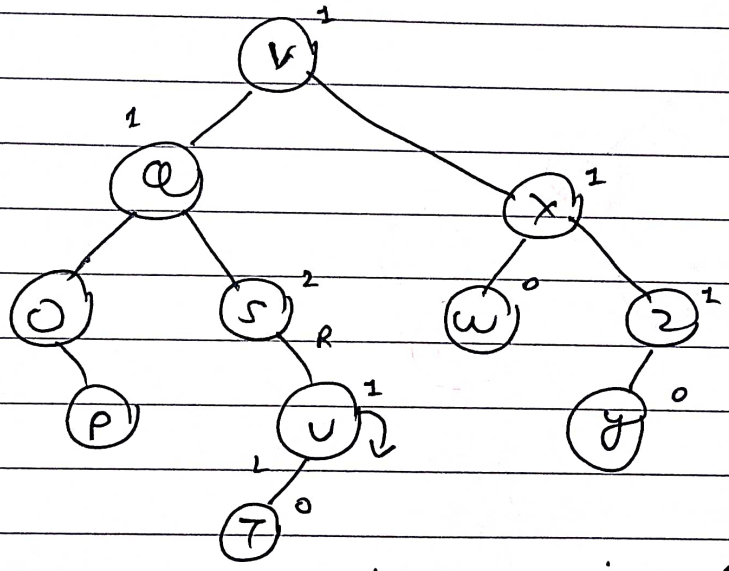
Topic.....Date.....



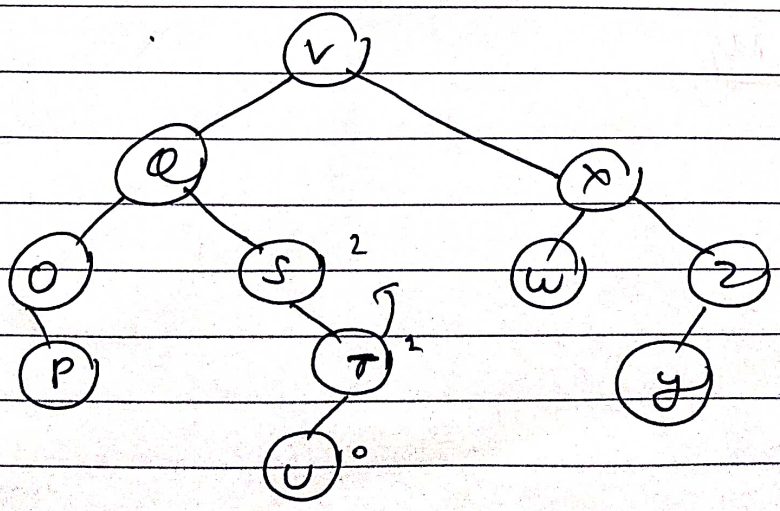
Balanced

ii) insert T

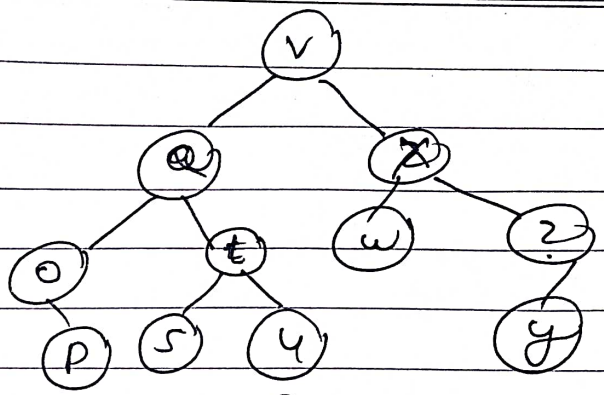
~~x) insert~~



unBalanced doing RL rotation

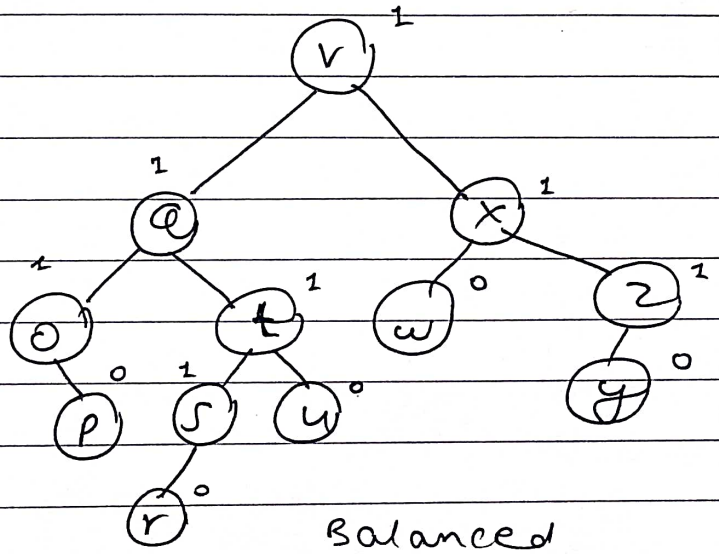


Topic.....Date.....



Balanced

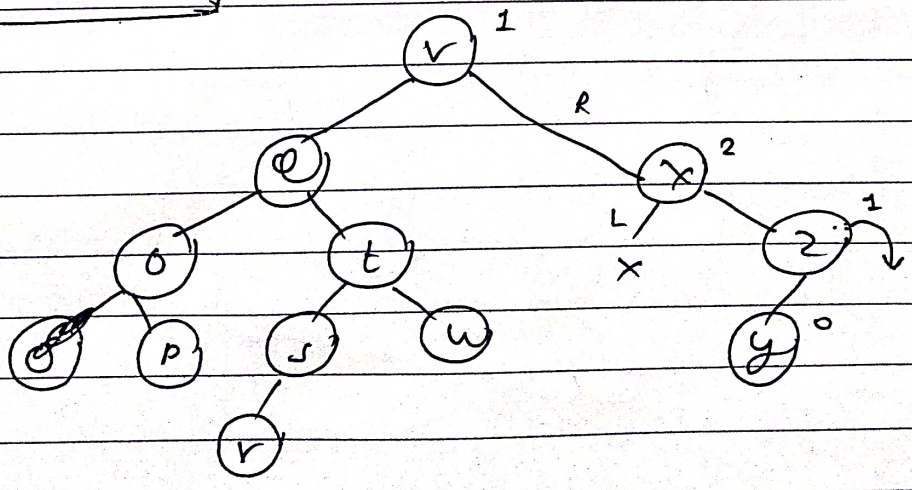
x) insert R



Balanced

which is required AVL tree.

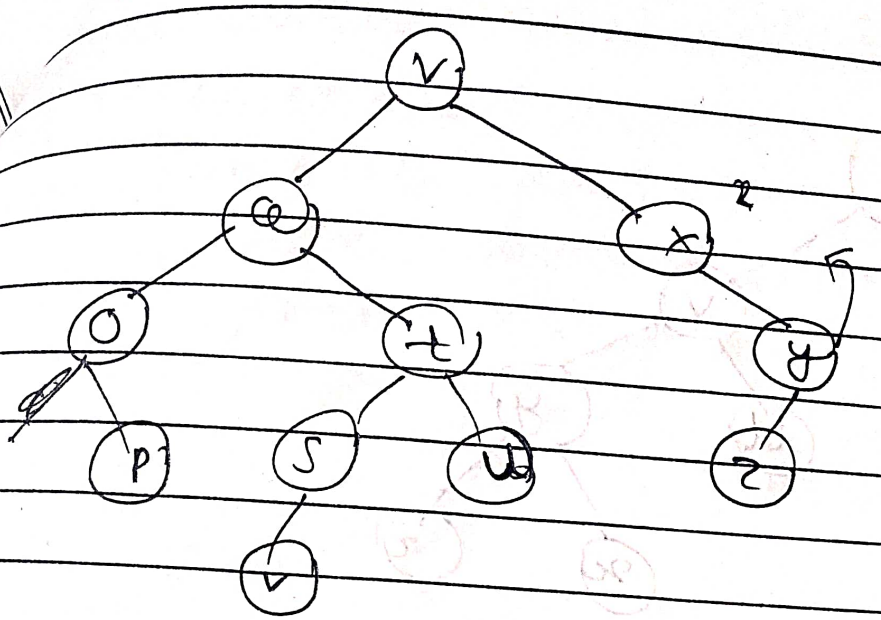
To delete BV

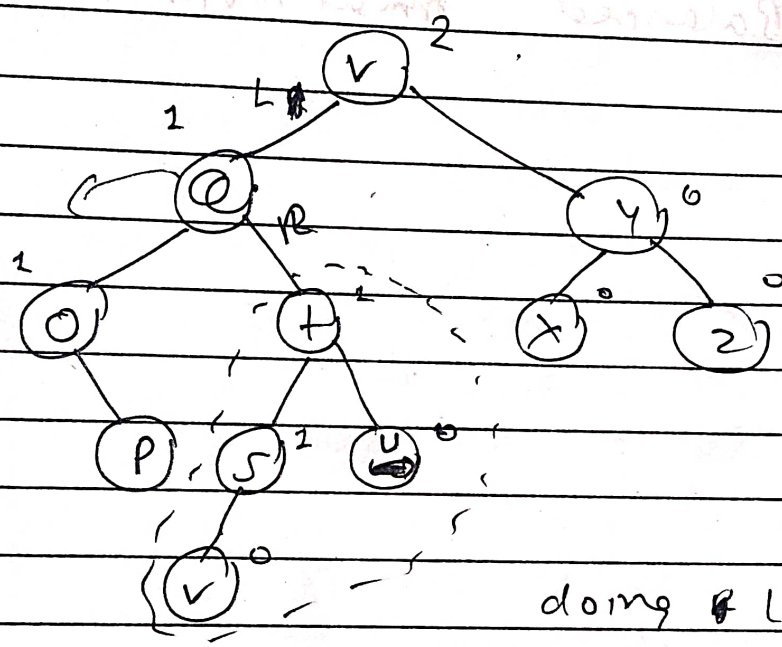


right ~~subtree~~ child left subtree deleted.

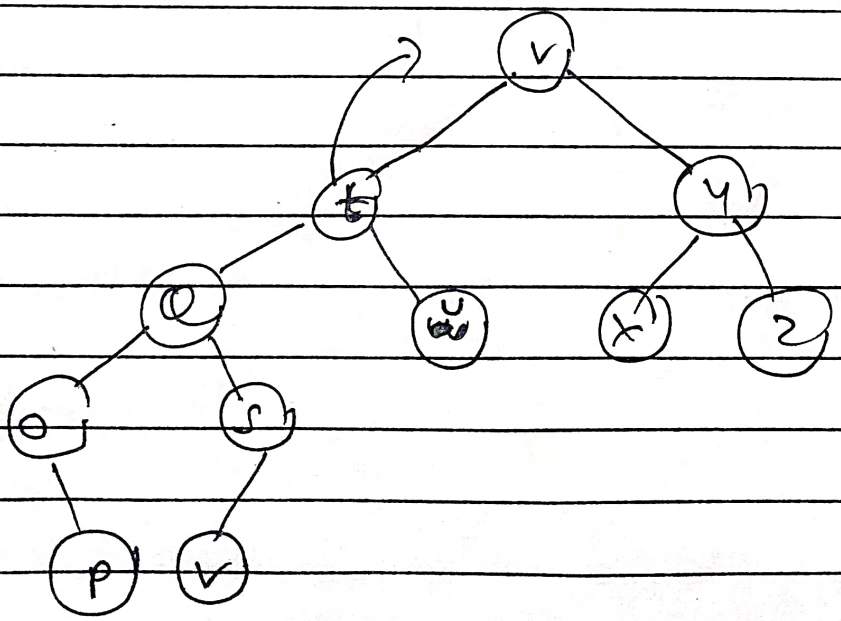
doing RL rotation

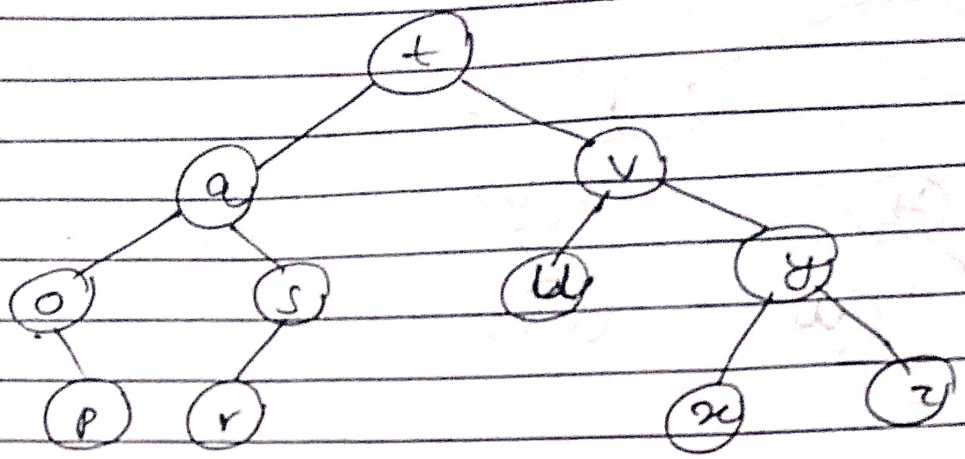
Teacher's Sign.....





doing LR rotation





Balanced Binary AVL tree