Solution Set and Rubric for Grading for Quiz 1

Question 1. Consider a binary tree with labels such that the pre-order traversal of the tree lists the elements in increasing order. Let us call such a tree a pre-order search tree. The tree is pointed to by a variable T.
Each node of T has the following fields:

- Left : points to left subtree (may be null)
- Right: points to right subtree (may be null)
- Value: the value stored at that node (may be null)

Assume that all values are distinct.

Question 1A. Write pseudo-code to for find(T,v) which finds the value v appears in the pre-order search tree pointed to by T. Show using an example. Marks 2.

Solution.

```
function Q=find(T,v)

  if T==null return(T); end;
  if T.value==v return(T); end;  // found
  if ((T.right!=null && v>=(T.right).value) Q=find(T.right,v); return(Q); end; // must in T.right
  if ((T.left!=null) && v>=(T.left).value) Q=find(T.left,v); return(Q); end; // or else in T.left (order
important)
  return(null); // then not found

endfunction;
```

Stuff	Marks
Almost Correct Code	1
Example	1

Question 1B. Write pseudo-code to pred(T,v) to find the predecessor of the value v in the pre-order search tree pointed to by T. Show using an example. Marks 3.

Solution.

```
function V=find_max(T)
// finds the maximum value in the pre-order search tree
  if T==null return(T); end;
  if (T.right!=null) return(find_max(T.right)); end; // right if non-null
  if (T.left!=null) return(find_max(T.left)); end; // or else left
  return(T.value); // finally the root (order important)

endfunction;


function Q=pred(T,v)

  if T==null return(null); end;
  if T.value==v return(null); end; // if at root, then no pred
  if ((T.left!=null && v==(T.left).value) return(T.value); end; //check left root, easy
  if ((T.right!=null && v==(T.right).value)   // right root: tricky
          if (T.left!=null) return(find_max(T.left)); end;
          else return(T.value); end;
// now we know it is not one of the roots
  if ((T.right!=null) && v>(T.right).value) Q=pred(T.right,v); return(Q); end;
  if ((T.left!=null) && v>(T.left).value) Q=pred(T.left,v); return(Q); end; // order important
  return(null); // finally

endfunction;
```

Stuff	**Marks**
Almost Correct Code	2
Example	1

Question 1C. Write pseudo-code to delete(T,v) to delete the value v in the pre-order search tree pointed to by T. Show using an example. Marks 3.

Solution.

```
function Q=merge(Tl,Tr)
// function to delete the root and return a tree with the deleted root
// if one of the childs are null then easy
 if (Tl==null) return(Tr); end;
  if (Tr==null) return(Tl); end;
// else
  Tree merged_tree;
  merged_tree.value=Tl.value; // hike up the left value
  merged_tree.right=Tr;
  merged_tree.left=merge(Tl.left,Tl.right); // now discard the root of Tl and go down Tl
  return(merged_tree);
```

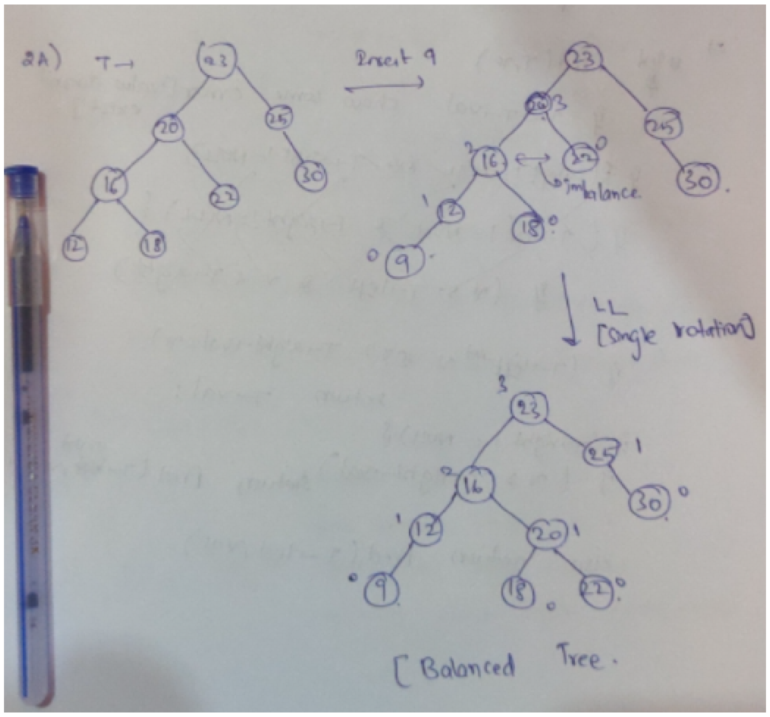
```
endfunction;

function Q=delete(T,v)

  if (T==null) return(T); end;
  if (T.value==v) return(merge(T.left,T.right); end; // the main case - value is found in the root
  if ((T.right!=null && v>=(T.right).value) T.right=delete(T.right,v); return(T); end; // else right first
  if ((T.left!=null) && v>=(T.left).value) T.left=delete(T.left,v); return(T); end; // or left , in order
  return(null);

endfunction;
```

There are other solutions as well which merge the two into one function.

Stuff	**Marks**
Almost Correct Code	2
Example	1

Question 2A. Insert the value 9 in the following AVL tree T and re-balance if required. Show all the steps and the final tree. Marks 3.

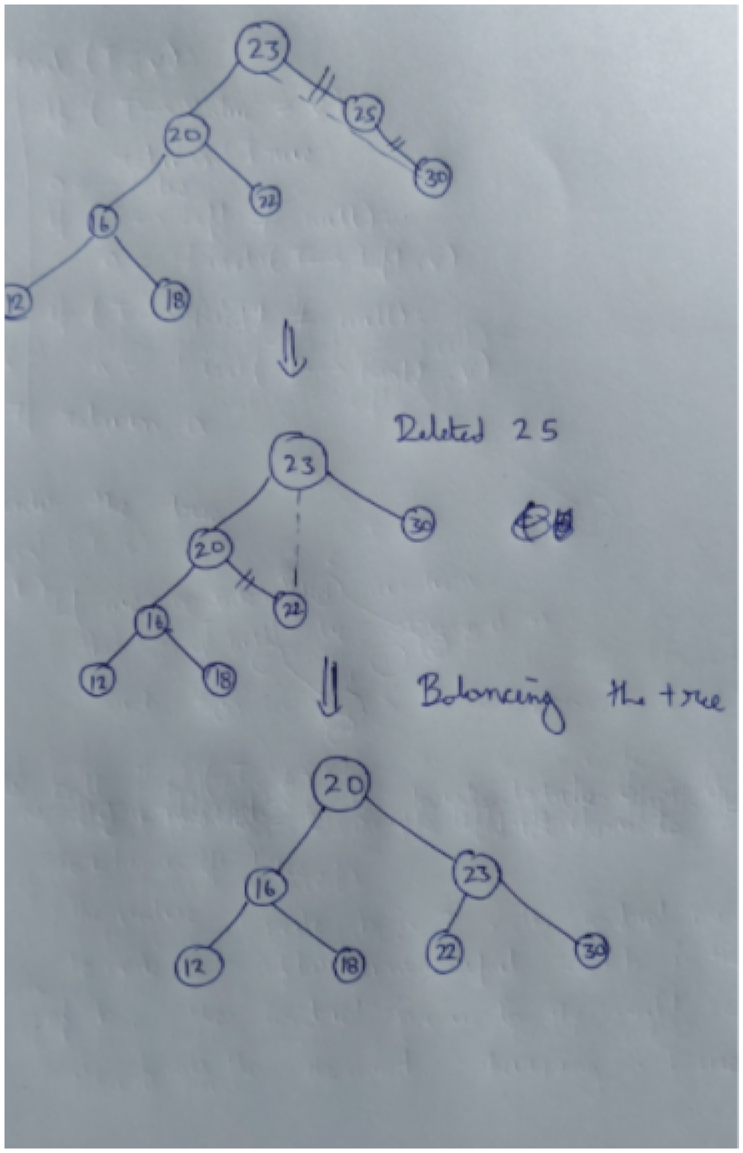
Solution.



Rubric:
Correct insertion of Node 9: 1 marks
Correct balanced BST using proper rotation: 2 marks

Question 2B. Delete the value 25 from the same AVL tree T (i.e., before the addition of 9) given in question 2A. Use replacement logic as in BST, and rebalance if required. Show all the steps and display the final tree. Marks 3.

Solution.



Rubric:
Correct deletion of Node 25: 1 marks
Correct balanced BST after proper rotation: 2 marks

Question 3. We have seen the mess table queue problem in which students join the mess queue behind the last person from their hostel, or at the end of the queue, if there is no one from their hostel. We have seen that the average waiting time does not change. However, is the average waiting time the same for students of different hostels? If so, why? If not, why not? Argue for a counter-example. Marks 3.

Solution. In general, the average waiting time for students of different hostels is not the same. We can show this via the following simple counter-example.

Suppose there are two hostels, namely H1 and H2. For simplicity, let H1 have only one student, named P1, and H2 have only two students, named P2 and P3. It is reasonable to assume that the order in which the three students show up to join the mess queue is uniformly random; i.e., the 3! = 6 permutations of their arrival are equally likely. We then compare the expected waiting times for P1, a student of H1, and P2, a student of H2.

First, consider P1. There are 6 possible arrival orders of the 3 students. For each arrival order, we observe the position in the queue that is allotted to P1, considering P2/P3 are allowed to join behind each other even if one of them arrives after P1.

Arrival Order of Students -> Position of P1 in Queue

P1, P2, P3 -> 1
P1, P3, P2 -> 1
P2, P1, P3 -> 3
P2, P3, P1 -> 3
P3, P1, P2 -> 3
P3, P2, P1 -> 3

Clearly, if P1 does not arrive first, he gets placed at the end of the queue. So, for 2 orderings, P1 has the first position (and thus a waiting time of 0), and for 4 orderings, P1 has the third position (and thus a waiting time of 2).

Expected waiting time for P1 = (2x0 + 4x2) / 6 = 4/3.

Now, we analyze P2 similarly, observing the position in the queue allotted to P2 for each possible arrival order.

Arrival Order of Students -> Position of P2 in Queue

P1, P2, P3 -> 2
P1, P3, P2 -> 3
P2, P1, P3 -> 1
P2, P3, P1 -> 1
P3, P1, P2 -> 2
P3, P2, P1 -> 2

So, P2 has first place (wait time = 0) for 2 orderings, second place (wait time = 1) for 3 orderings, and third place (wait time = 2) for 1 ordering.

Expected waiting time for P2 = (2x0 + 3x1 + 1x2) / 6 = 5/6.

As we can see, the expected waiting times for students of different hostels are not the same. This happens because of an asymmetric student distribution among the hostels; students from less populated hostels would have to wait longer on average, since it is less easy for them to skip ahead in the queue as compared to students from more populated hostels. If the number of students in each hostel was the same, the average waiting times would be the same too, by symmetry.

Stuff	**Marks**
Right Guess	1
Right argument	1
Some example	1