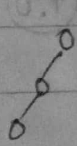
Raghav mittal
1BM18CS075

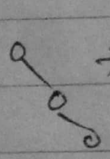
Insertion & deletion of AVL Tree..

Insertion

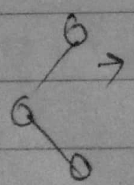
```
node* insert (node* node , int key)
    {
        if (node = = NULL)
            return (newnode (key));

        if (key < node -> key)
            node -> left = insert (node -> left, key);

        else if (key > node -> key)
            node -> right = insert (node -> right, key)
            else
                return ;

        Calculate balance factor :-
```



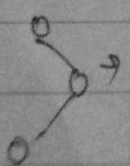
```
←   if (balance > 1 && key < node -> left -> key)
        return rightRotate (node);
```



```
→   if (balance < -1 && key > node -> right -> key)
        return left rotate (node);
```



```
                                          left
→   if (balance > 1 && key < node -> ~~right~~ -> key)
    { node -> left = leftRotate (node -> left)
        return rightRotate (node)
```



```
→   if ( balance < -1 && key < node -> right -> key
    node -> ~~[illegible]~~ right = rightRotate ( node -> right)
        return leftRotate (node).
```

Deletion:

```
node * delete (node* root, int key)
 {
  if (root == NULL)
      return root;
  if (key < root -> key)
      root -> left = delete node (root -> left, key)

  else if (key > root -> key)
      root -> right = delete node (root -> right,
                                             key)
Balance factor:

  if (balance >1 && ~~getB~~ getBalance (root -> left)) >=0
      return right Rotate (root);

  if (balance >1 && getBalance (root -> left )<0)
     {
      root -> left = left Rotate (root -> left).
      return. rightRotate (root);

  ~~if (balance <-1 && getB~~
     if (balance < -1) && getBalance (root -> right
                                              <=0)
        return left Rotate (root);
  if ( balance < -1 && getBalance (root -> right)
                                              >0)
 { root -> right = rightBalance (root -> right);
     return leftRotate (root);
 }.
```