Pseudo tode:

num Color (RED, BLACK) struct Node class RB Tree Junction cells, RBTree: : votate left (Node + d root, Node + apt) Void 1 Node *pt-right = pt -> right. pt -> right = pt=right > left; it lpt -> right ! = NULL pt->right -> parent = pt-; pt -> right -> porent = pt -> parent; if (pl-) parent == NULL) rod = pt-right; Obe it (p1 -= pl -> parent > left / Pl-sparent sleft= pl-right. else pt > powert = cight = pt - vight: pt-right-> left-pt-;
pt-right;

Scanned with CamScanner

```
RBIree: colate Right (Node * 4 root, Node Zept)
void
       Node * pt-left=pt-left;
          ph-sleft=ph-left-> right;
           it(pl-) left! = NULL)
                  pl-> left-> parent = pl;
          All pt-left parent: pt-parent;
               It (pt -> parent == NULL)
                        root: pt-left;
                else it ( pt == pt -> parent -> laft)
                          pt-> ponent -> left=pt-left:
                       pl- -> parent -> right = pt-left;
                else
                prostor pl-legt -> right = pt;
                       pt->parent = pt-left;
       RBTIPP: fix Violation ( Node * front, Node * Spr)
void
                Null * parent-pt = NULL;
                Node & grand-parent-pt = NULL;
                 while(Ept-1=root) de (pt->dor!=BLACK) 44
                          (pt-sparent-s color == REDI)
```

2

```
parent-A-=pl-> parent;
 grand-parent-pt=pl-) parent-) parent-;
 if (parent-pt = 2 grand-parent-pt -> left) MANNE (Selection)
        Node +uncle-pt = grand-parent-pt = sright;
        it (unde-pt- 1 = NULL el unde-pt- -> color = = RED)
              grand-potent-At > color = RED;
               parent-pt -> idor = BLACK;
                unde-pl-> color = BLACK;
                 pl-: grand-parent-pt-;
              MANNE SANDERS SANDER
              if (pt == palent-pt -> right)
                    rotate legh- (root, palent-pt-)
                       ot = parent-pt;
                        parent-pt= pt > parent;
                y
                rotate Right (root, grand-parent-pl-);
                 swap (parent-pt -> color, grand-parent-pt -sador)
                 pl== povert-pl=;
```

```
else
      Node & unitest= grand-parent-pt-> left:
           if ((unde pt != NULL) && (uncle pt -> volor = = RED))
                  grand-pared-11- ) whor = REO;
                  portent-pl- -> color =BLACK;
                    unde_pl-) udor=BLACK;
                     pl = grandparent-pl-;
           else
                   if (pt == powert pt sleft)

rotate Right (root, pavent-15-);
                            pt= parent-pt=;
                            parent-pt= pt->parent;
                    rotate left (root, grand-patent-pt);
                      Swap ( parent-pl-> color, grand-parent-pl-> (olar).
                        pt= parent- pt;
      rust -> color = BLACK;
```

void estree: insert (wost- int- Without)

Node # Pl= new Node (n);

(ool-= BSTinsert (root, Pl-);

his Violation (root-, pl-);