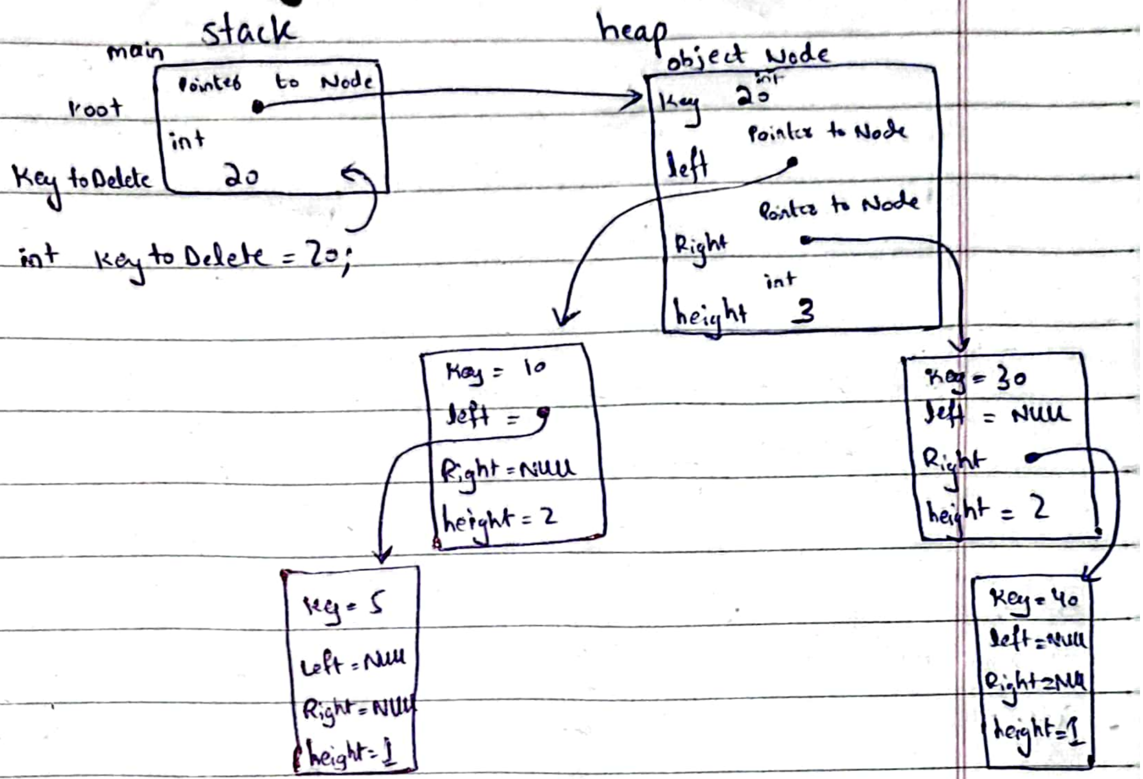
Node* return a pointer to the node

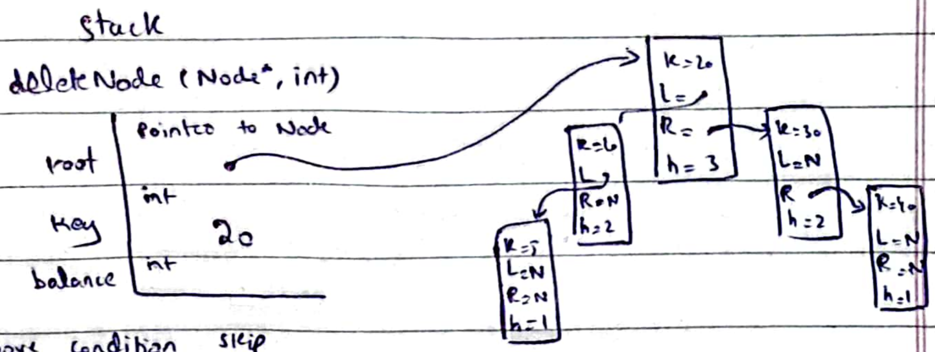
Day: ______ Date: ______

Dry Run Deletion



int KeytoDelete = 20;

root = deleteNode (root, KeytoDelete) function call.



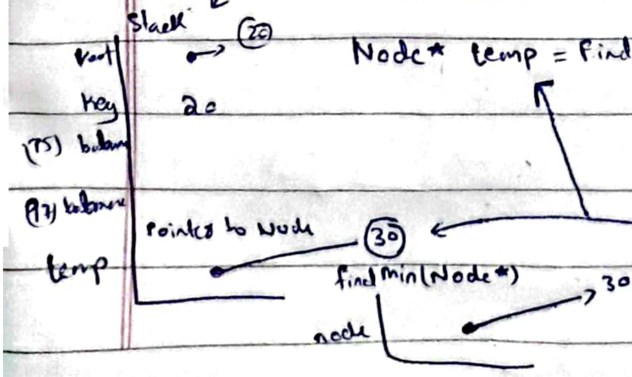
above condition skip

else{

# Node with two children

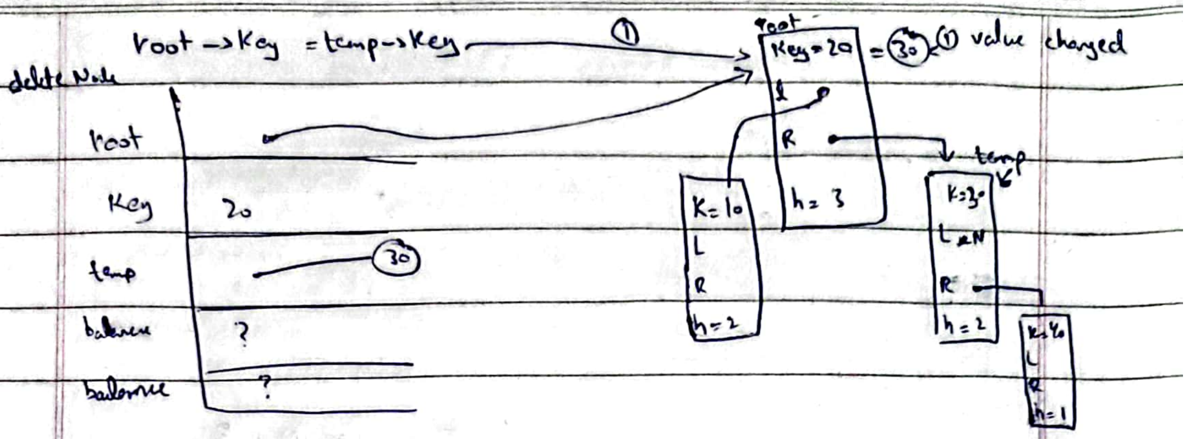
```
Node* temp = Findmin (root→right)
```

Jump to function

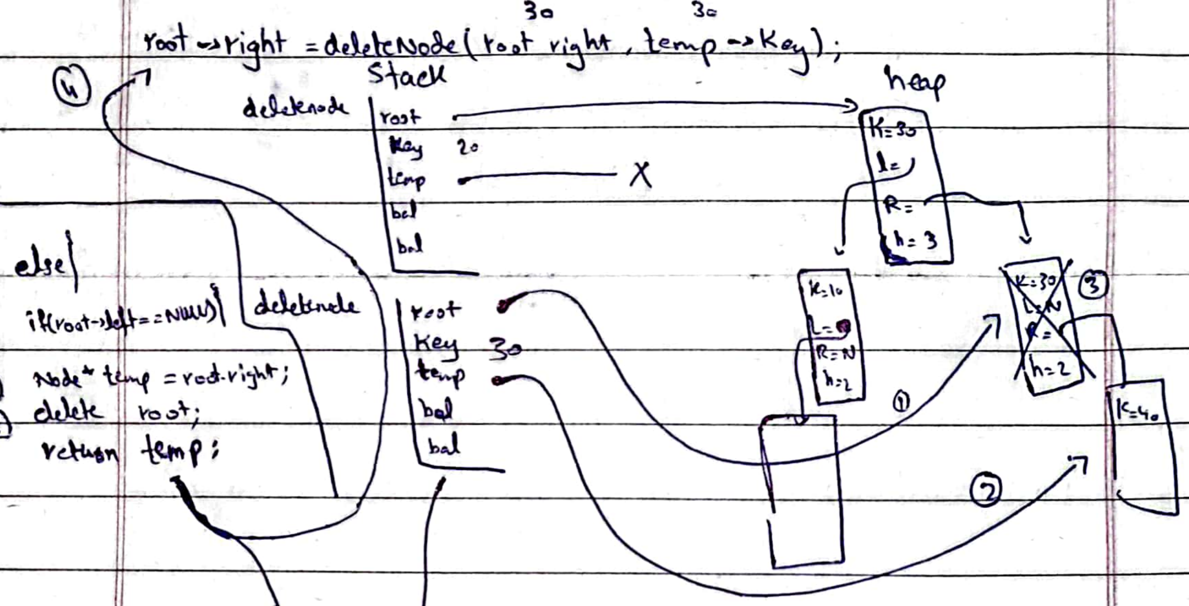


```
Node* findmin (Node* node){
skip while (node->left != NULL)
return Node;
```

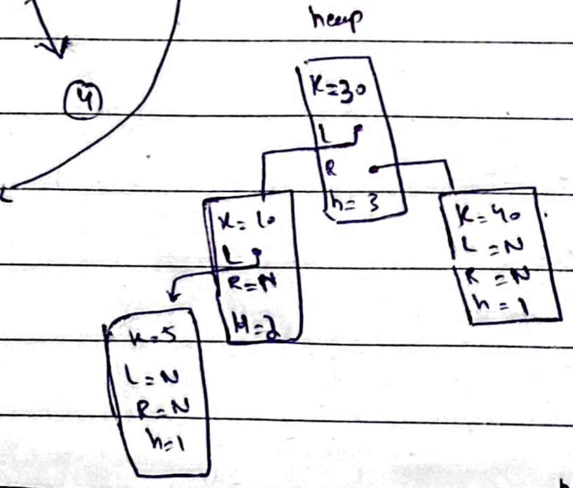
```
root -> Key = temp->Key
```



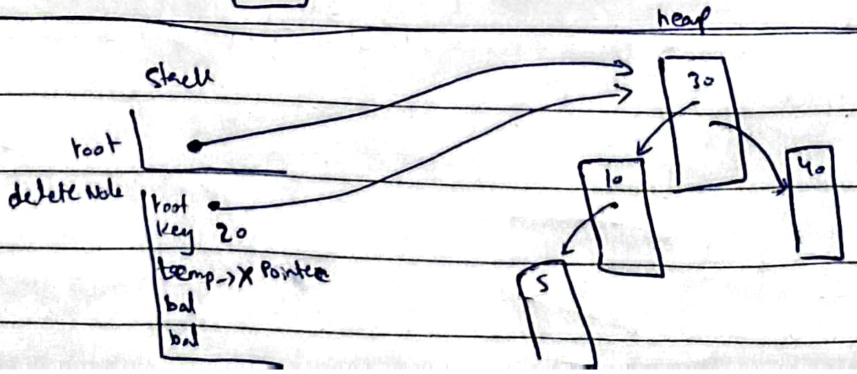
```
        30                        30
root->right = deleteNode( root right , temp->Key);
```



```
else{
  if(root->left==NULL){
② Node* temp = root.right;
③ delete root;
  return temp;
```



This will end after ④



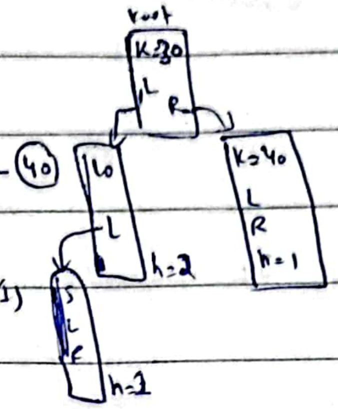
root->height = 1 + max(height(root->left), height(root->right));

height(root->right

node | → 40

return node->height; (1)

⋮



1+ max (2, 1)

max (int,int

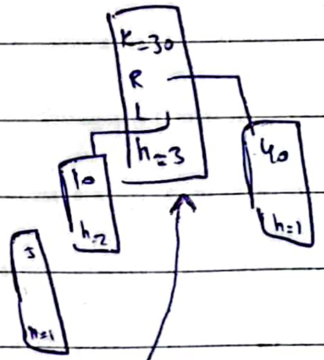
a | 2

b | 1

return 2)

root->height = 1+2 = 3

~~int~~ int balance = getbalance(root)
(30)



~~int~~ getbalance(Node*)

node |

return height(node->left) - height(node->right)
2-1 => 1
Same
height for both

node | → 30

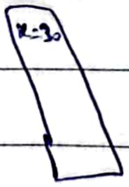
deletenode

root |

key | 20

temp | X

bal | 1



no condition will run because tree is balance.