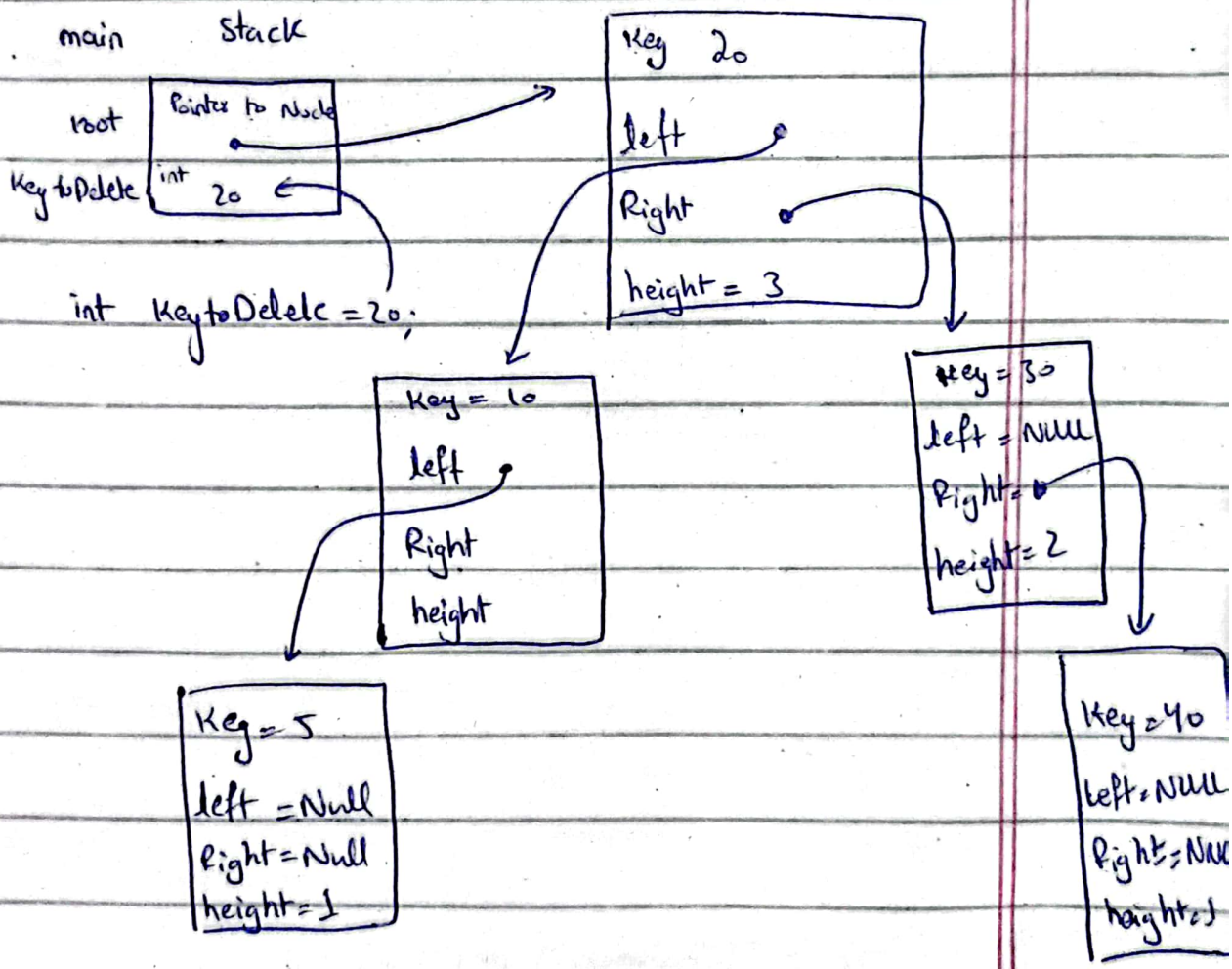
Name :-

Muhammad - Zamas

Reg No :-

SP22 - BCS - 061

Dry Run Deletion..



root = deleteNode(root, KeytoDelete) function call

20 20

Stack

deleteNode (Node*, int

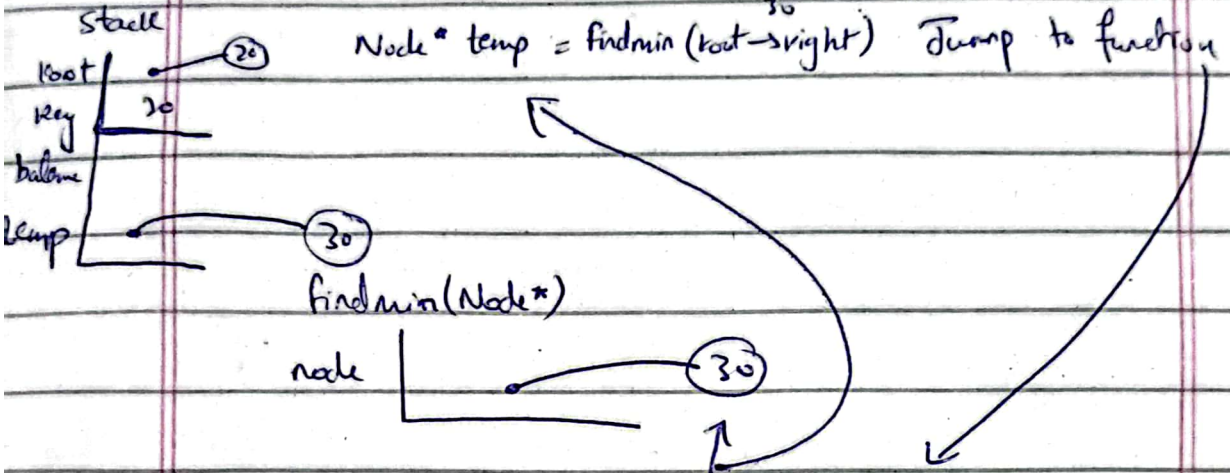
root	Pointer to node
Key	int 20
balance	int

above condition skip

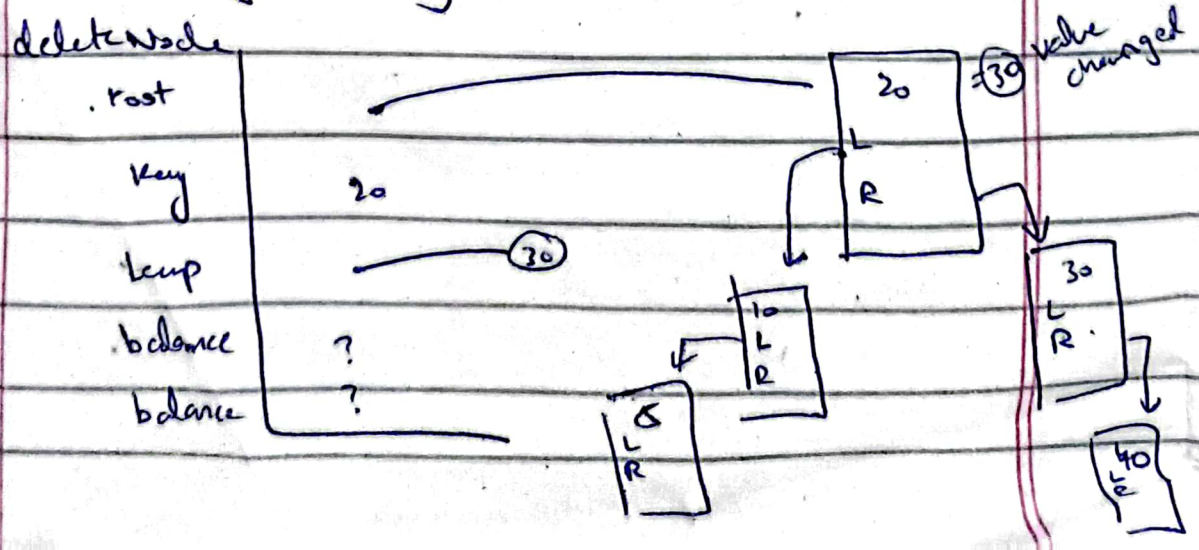
else {

#Node with two children

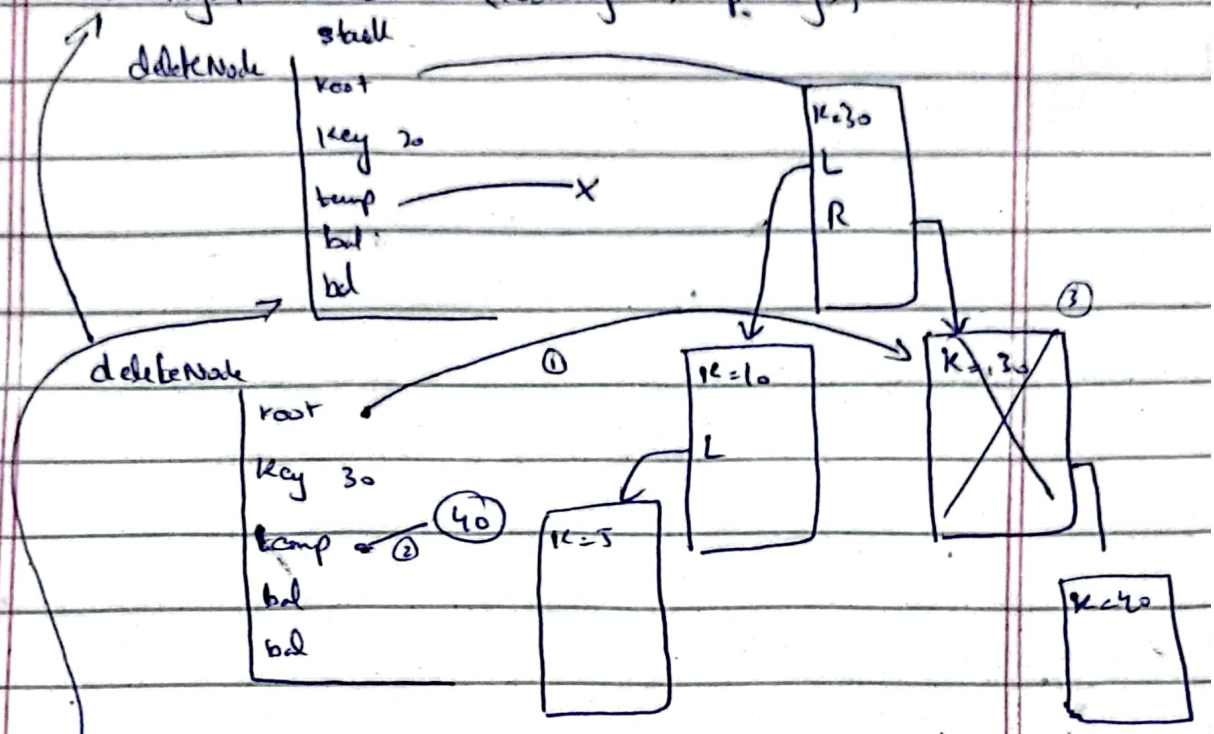
Node* temp = findmin (root->right) Jump to function



root->Key = temp Key



root->right = deleteNode(root->right, temp->key);



else {

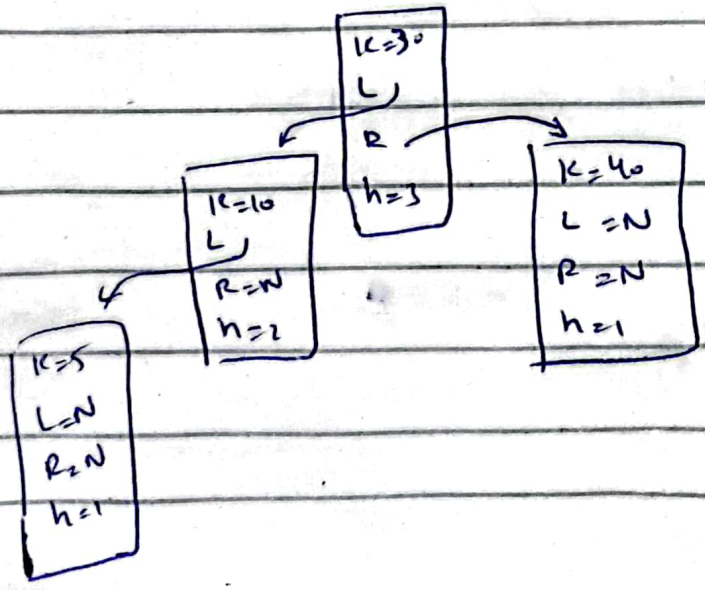
if (root->left == NULL) {

② Node* temp = root->right.

③ delete root

④ return temp;

what left in heap



Insertion Day 1 Run

```
int main() {
```

stack

main | Pointer to Node
root | NULL

Node* root = NULL;

```
root = insert (root, 10)
```

↓

function called

stack

```
insert (Node*, int)
```

Pointer to Node
root | NULL
key | 10

```
if (root == NULL)
```

condition true

```
return new Node(key);
```

↓ heap object Node

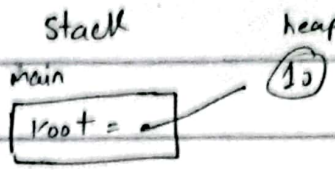
key = 10
left = NULL
Right = NULL
height = 1

all condition will be skip

return root;

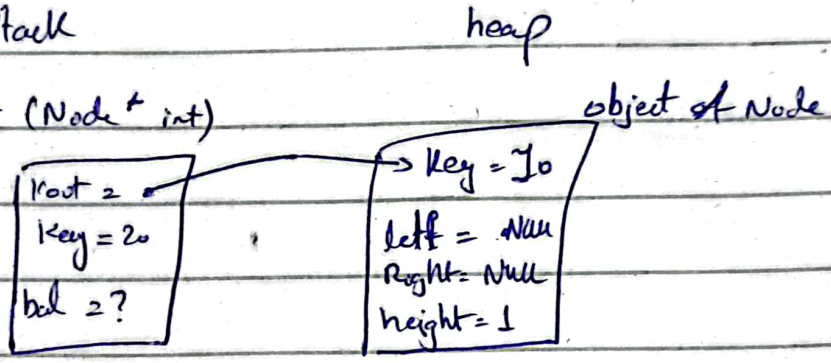
↓

of insert function



root = insert (root, 20) function called.

Node* insert (Node * root, int Key)



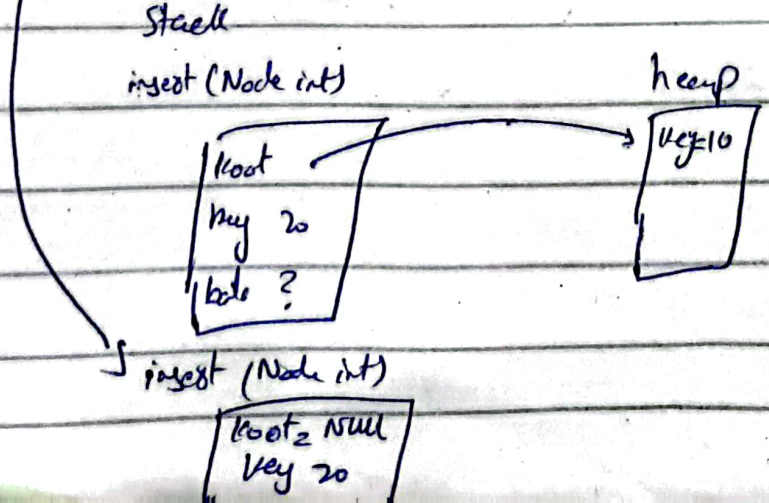
condition skips

else if (key > root->key)

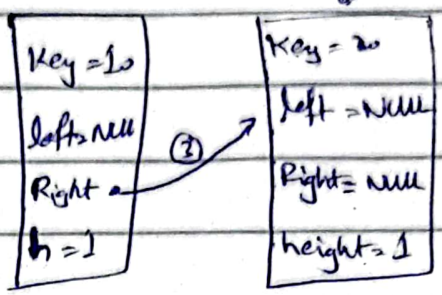
root->right = insert(root->right, key); recursively

↓

Null



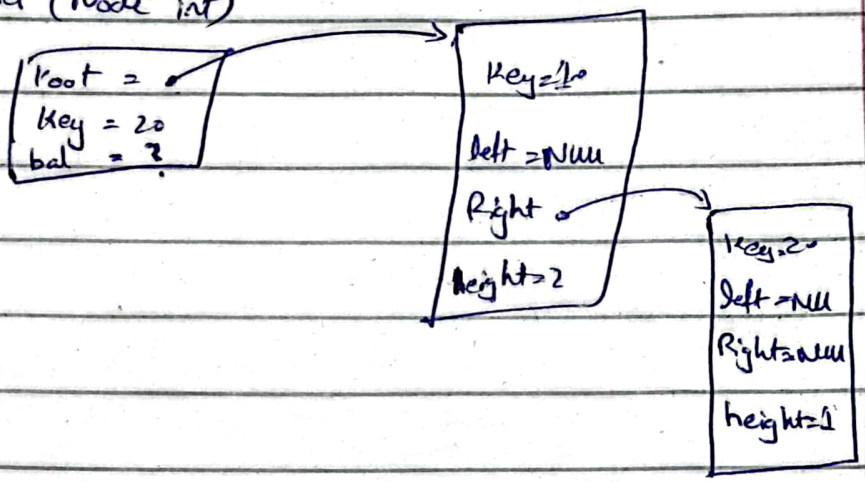
if (root == NULL)
    return new Node(key);



③ because we store it in root->right

Stack updated

insert (Node* int)



root -> height = 1 + max(height(root->left), height(root->right));
(0, 1)

Node == NULL (false condition)

return Node -> height => (1) in this case

for height(root->left) which is NULL so return 0.

Stack

new (int, int)
a = 0
b = 1

a > b ? a : b; return b = 1 in this.

root height = 1+1 = 2

get balance (root);

stack

main

root → (10)

insert (Node* int)

root = → (10)

key = 20

bf = -1

return (height (node->left) - height (node->right));
(0-1)

next condition will be skip

heap

Key 10
Left = NULL
Right →
height = 2

Key 20
left = NULL
Right = NULL
height = 1