insertion Dry Run.

Day: _____ Date: _____

```
int main() {
```

stack

main

Pointer to node

root | NULL

```
Node * root = NULL;
```

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

↓

function called.

stack

insert (Node*, int)

root — Pointer to node — NULL

Key | 10

```
if (root == NULL)        condition true
    return new Node(Key);
```

heap object Node.

Key = 10

left = NULL

Pointer to node

right = NULL

height = 1

all condition will be skip

```
return root;
```

↓

of insert function.

stack

main

root =

```
root = insert (root, 20) function called.
Node* insert (Node* root, int Key)
```

stack

insert (Node*, int)

root =

Key = 20

~~height~~

~~next element kon case me~~ figure out hogi.

heap

object Node 0

Key = 10

left = NULL

right = NULL

height = 1

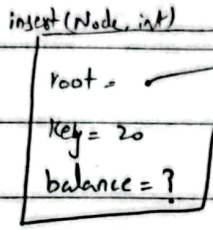
balance factor = bf.

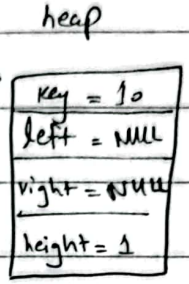
Day: condition skip Date:

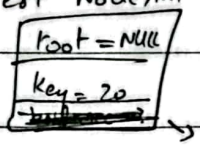
else if (key > root->key) time...

root->right = insert (root->right, key); NULL recursively call itself mean.

stack



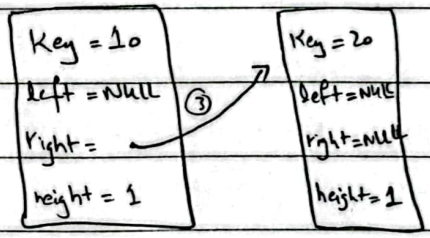




if (root = NULL)
return new Node(key);

means

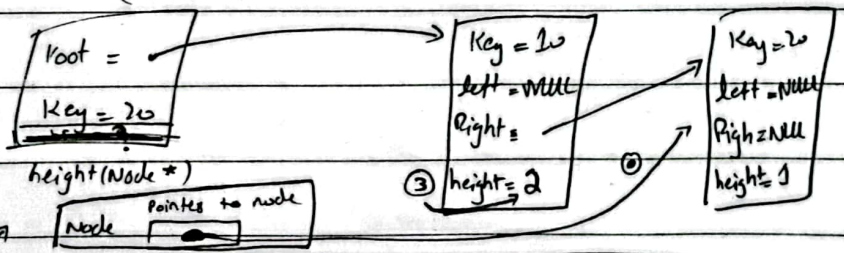
because we store it in root right



As new node inserted updated stack

insert root = NULL key = 20 end or remove from stack.

insert (Node*, int)



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

Node == NULL (false condition)

return ~~right~~ node->height => (1) in this case.

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

stack

max (int, int)

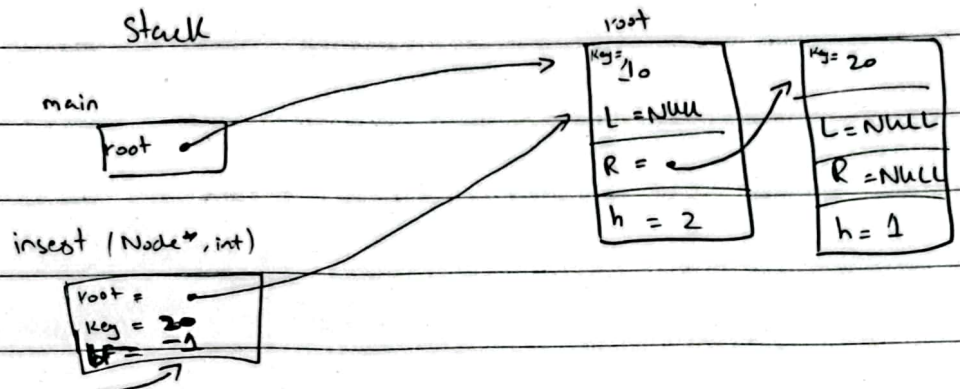
a = 0
b = 1

(a > b) ? a : b ; return b = 1

Day: _____ Date: _____

root->height = 1 + 1 = 2.

get balance(root);



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

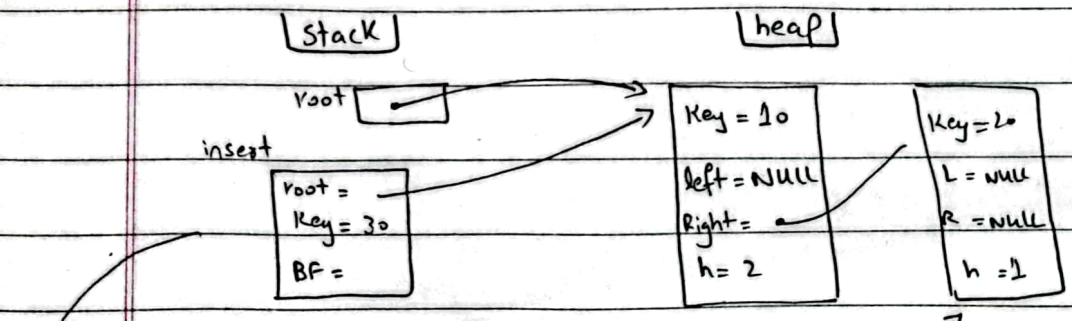
same as before calculating height;

next condition will be skip.

⋮

Third insert

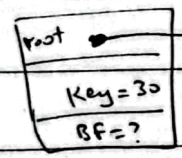
root = insert(root, 30);



else if (key > root->key)

root->right = insert(root->right, key) (20, 30)

for this recursive function

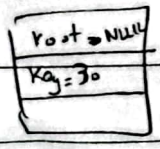


① other condition skip

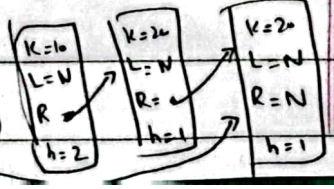
else if (key > root->key)

root->right = insert(root->right, key) (NULL, 30)

20 ka right



first condition new Node(key)



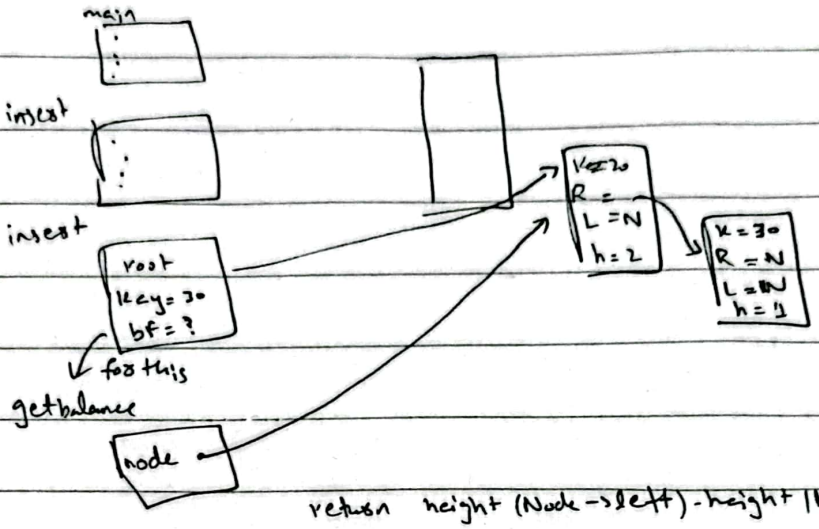
Day: Now for Node 20 Date:

we check it height.

root->height = 1 + max(height(root .....)

↳ (20)

same as we done with in 10 last insertion ma...



return height(Node->left) - height(Node->Right)

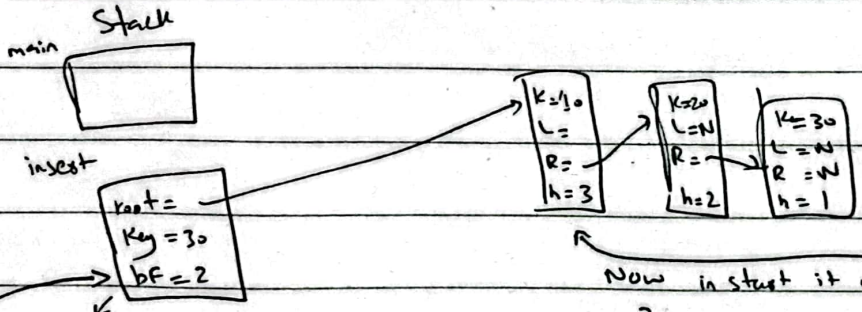
NULL - 1

0 - 1

-1

next condition for 20 will skip

return to first time we call insert.



Now in stack it was 2.

where we left

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

Same procedure

1 + max(0, 2);

= 1 + 2 = 3

get balance height(root left) - height(root Right)

0 - 2 = 2

Now rotation ~~left~~ condition will True.

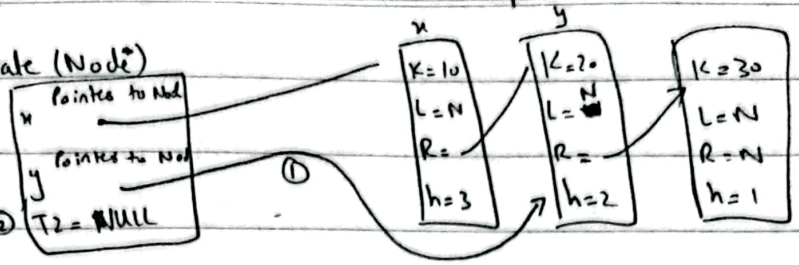
if Right Right case.

if (balance > 1 && key > root->Right->key)

return leftRotate(root); function call.

Day: Stack heap Date:

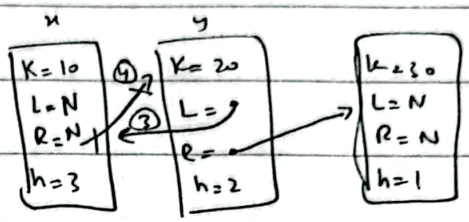
leftRotate (Node*)



① Node* y = x->Right;

② Node* T2 = y->left;

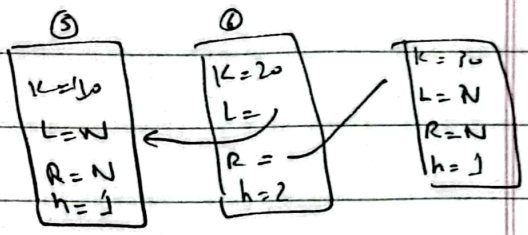
③ y->left = x



④ x->Right = T2

⑤ x -> height = ... calculating height

⑥ y->height = ... calculating height

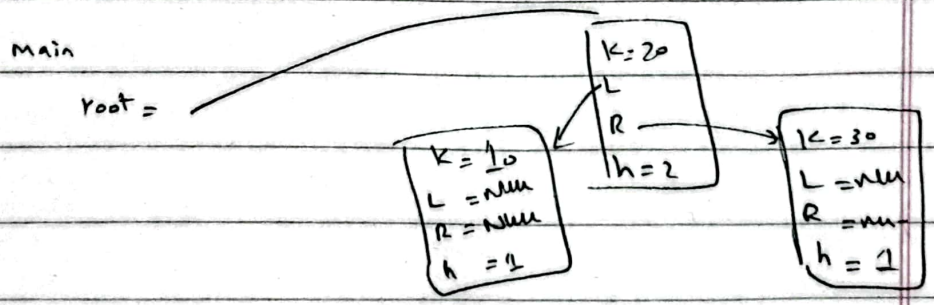


return y.

return root;

in main function

root = 20



rest will be same

add 40 and 5 taka easy raha sab ka liya ... 😊