Let us write Recursion Report to show that inorder function, called for BST depicted below, will print the numbers sorted in ascending order.

void inorder(Node \*x)

{

if(x!=NULL)

{

inorder(x->left) cout<<x->data<<endl; inorder(x->right)

}

}

28

23

45

15

25

35

56

# Report 1

Function call: x points to the node with 28

* inorder(x->left) // x points to the node with 23
* print 28;
* inorder(x->right) // x points to the node with 45

# Report 2

Rewriting the shaded part in Report 1 by calling the function with x pointing to 23

* inorder(x->left) // x points to the node with 15
* print 23;
* inorder(x->right) // x points to the node with 25 • print 28;
* inorder(x->right) // x points to the node with 45

Nodes with 15 and 25 are leaves, thus a function call for them requires just printing the data. Therefore, replacing

inorder(x->left) // x points to the node with 15

by printing 15 and

inorder(x->right) // x points to the node with 25

by printing 25 we get

* print 15
* print 23; • print 25
* print 28;
* inorder(x->right) // x points to the node with 45 **Report 3**

Rewriting the shaded part in Report 1 by calling the function with x pointing to 45

* print 15
* print 23; • print 25
* print 28;
* inorder(x->left) // x points to the node with 35
* print 45;
* inorder(x->right) // x points to the node with 56

Nodes with 35 and 56 are leaves, thus a function call for them requires just printing the data. Therefore, replacing

inorder(x->left) // x points to the node with 35

by printing 35 and

inorder(x->right) // x points to the node with 56

# by printing 56, we get the output with numbers sorted in ascending order

* print 15
* print 23; • print 25
* print 28;
* print 35
* print 45;
* print 56;

# Please submit your Recursion Report for the following BST

45

20

50

12

38

10

15

* Inorder(45)
* Inorder(20)
* Inorder(12)
* Inorder(10)
* Print 10
* Print 12
* Inorder(15)
* Print 15
* Print 20
* Inorder(38)
* Print(38)
* Print 45
* Inorder(50)
* Print 50