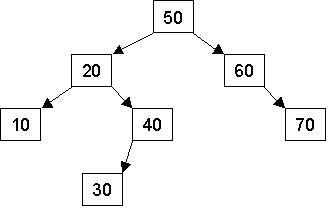
1. a)



25

39

15

71

80

65

b) PreOrder: 50 20 10 15 40 30 25 39 60 70 65 80 71

InOrder: 10 15 20 25 30 39 40 50 60 65 70 71 80

PostOrder: 15 10 25 39 30 40 20 65 71 80 70 60 50

c) **AFTER DELETING 30**

50

60

20

10

40

70

25

80

65

71

15

39

**AFTER DELETING 20**

50

60

15

10

40

70

25

80

65

71

39

1. a)

struct Node

{

Node(Node\* p , int val) : parent(p), Data(val), left(nullptr), right(nullptr)

{

}

int Data;

Node\* parent;

Node\* left;

Node\* right;

};

b) void insert(Node\* current, int value)

{

insertHelper(current, value, nullptr) *//recursively calls insert*

}

void insertHelper(Node\* current, int value, Node\* parent)

{

*If current is equal to nullptr*

*Create a new node at current, setting data to value and the parent pointer pointing towards itself*

*Else If value is less than current->data*

*Recursively call the function with parameters current->left, value, current*

*Else if value is greater than current->data*

*Recursively call the function with parameters current->right, value, current*

}

1. a)

6

4

2

0

3

7

b) Array (from first to last element)

7 3 6 0 2 4 //Size = 6; arr[0] = 7,…,arr[5] = 4

c) After using remove once more:

6 3 4 0 2 //Size = 5; arr[0] = 6,…,arr[4] = 2

1. a) O(C + S)

b) O(logC + S)

c) O(logC + logS)

d) O(logS)

e) O(1)

f) O(logC + S)

g)O(SlogS)

h)O(C\*logS)