HOMEWORK 5

1. A.

Adding 80

50

20 60

10 40 70

30 80

Adding 65

50

20 60

10 40 70

30 65 80

Adding 75

50

20 60

10 40 70

30 65 80

75

Adding 15

50

20 60

10 40 70

15 30 65 80

75

Adding 35

50

20 60

10 40 70

15 30 65 80

35 75

Adding 25

50

20 60

10 40 65 70

15 30 80

25 35 75

b. (Using line breaks for clarity)

IN ORDER

10

15

20

25

30

35

40

50

65

60

70

75

80

PRE ORDER

50

20

10

15

40

30

25

35

60

65

70

80

75

POST ORDER

15

10

25

35

30

40

20

65

75

80

70

60

50

c.

Deleting 30

50

20 60

10 40 70

15 35 65 80

25 75

Deleting 20

50

15 60

10 40 70

35 65 80

25 75

1. A.

struct Node

{

int value;

Node\* leftChild;

Node\* rightChild;

Node\* parentNode;

};

B.

void insertNode(int dataToBeInserted)

{

//assuming pointer to the head of the BST is called headNode

Node\* puttingIn = new Node;

puttingIn -> value = dataToBeInserted;

puttingIn -> leftChild = nullptr;

puttingIn -> rightChild = nullptr;

puttingIn -> parentNode = nullptr;

if(headNode == nullptr)

{  
 headNode = puttingIn;

return ;

}

Node\* curentNode = headNode;

bool isOnLeft = false;

while(currentNode != nullptr)

{

if(currentNode -> value == dataToBeInserted)

{

delete puttingIn;

return;

}

if(currentNode -> value > dataToBeInserted)

{

isOnLeft = true;

puttingIn -> parentNode = currentNode;

currentNode = currentNode -> leftChild;

continue;

}

if(currentNode -> value < dataToBeInserted)

{

isOnLeft = true;

puttingIn -> parentNode = currentNode;

currentNode = currentNode -> rightChild;

continue;

}

}

currentNode = puttingIn;

if(isOnLeft)

puttingIn -> parentNode -> leftChild = currentNode;

else

puttingIn -> parentNode -> rightChild = currentNode;

}

3.

a.

8

3 6

0 2 4

b.

8 3 6 0 2 4

c.

6 3 4 0 2

4.

a.

O(C + S)

b.

O(log C + S)

c.

O(log C + log S)

d.

O(log S)

e.

O(1)

f.

O(log C + S)

g.

O(Slog S)

h.

O(C log S)