Homework 5 Report

**Problem 1A:**

50

60

70

80

71

65

20

10

40

30

25

39

15

**Problem 1B:**

inorder traversal:

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

pre-order traversal:

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

post-order traversal:

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

**Problem 1C:**

50

60

70

80

71

65

15

10

40

25

39

**Problem 2A:**

struct Node

{

Node (const int& data)

{

m\_value = data;

m\_leftChild = m\_rightChild = m\_parent = nullptr;

}

int m\_value;

Node \*m\_leftChild, \*m\_rightChild, \*m\_parent;

};

**Problem 2B:**

void insert(const int& myValue)

if the tree is empty

allocate a new node with myValue as its data

point the root pointer to the new node

start with a current pointer to the root of the tree

while we're not done

if myValue is equal to the current node's value, return

(because the value is already in the tree)

if myValue is less than the current node's value

if current node has a left child, point current to

that child

else

allocate a new node with myValue as its data

set current node's leftChild pointer to new

node

set new node's parent pointer to current node

return

if myValue is greater than the current node's value

if current node has a right child, point current to

that child

else

allocate a new node with myValue as its data

set current node's rightChild pointer to new

node

set new node's parent pointer to current node

return

(Note: above pseudocode was adapted from Carey Nachenberg’s powerpoint slides)

**Problem 3A:**

7

6

4

3

0

2

**Problem 3B:**

In an array, it would appear as follows, with the left-most box being index 0:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 7 | 3 | 6 | 0 | 2 | 4 |

with a counter variable of:

int num\_elements = 6

**Problem 3C:**

The resulting array would be the following, with the left-most box being index 0:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 3 | 4 | 0 | 2 |

with a counter variable of:

int num\_elements = 5

**Problem 4A:** O(C + S)

**Problem 4B:** O(log C + S)

**Problem 4C:** O( log(CS) )

**Problem 4D:** O(log S)

**Problem 4E:** O(1)

**Problem 4F:** O(log C + S)

**Problem 4G:** O(S log S)

**Problem 4H:** O(C log S)