1.Consider the following binary search tree, ordered with < relationship.

a. add 80, 65, 75, 45, 35, 25

50

20 60

10 40 70

30 45 65 80

25 35 75

b.delete 30 then 20

50

25 60

10 40 70

35 45 65 80

75

c. using tree in part a, what is post order, preorder, in order traversals?

Preorder (Print, left, right): 50, 20, 10, 40, 30, 25, 35, 45, 60, 70, 65, 80, 75

Inorder(left, print, right): 10, 20, 25, 30, 35, 40, 45, 50, 60, 65, 70, 75, 80

Postorder(left, right, print): 10, 25, 35, 30, 45, 40, 20, 65, 75, 80, 70, 60, 50

2. Consider the operations on an initially empty maxheap h ordered by < relationship

Insert: 3, 5, 2, 1, 10, 4

Delete: (10)

Insert: 8, 6

Delete (8)

a. show the resulting heap in tree form

6

3 5

1 2 4

b. show the resulting heap in array

{6, 3, 5, 1, 2, 4};

c. remove the largest item one more time and show the heap in array form.

{5, 3, 4, 1, 2};

3. In some BST, each node has a left, right, and parent pointer.

a. show a C++ structure for a node that has child ptrs and parent pointers

struct CareyNode

{

CareyNode() {leftcarey = nullptr; rightcarey = nullptr; parentcarey = nullptr;}

int careyvalue;

CareyNode\* leftcarey;

CareyNode\* rightcarey;

CareyNode\* parentcarey;

};

b. write pseudocode to insert a new node into this BST

void careyinsert(int value)//assuming the program already has a “root” pointer

{

CareyNode\* add = new CareyNode();

add->careyvalue = value;

if (tree is empty)

{

make root = add;

return;

}

CareyNode \* temp = root;

while (2)

{

if (value == temp->careyvalue)

return;

if (value < temp->careyvalue)

{

if(temp->left != nullptr)

make temp = it’s left node

else

{

set temp-> left to be add;

set add->parent = temp;

return;

}

}

else if (value > temp->careyvalue)

{

if(temp->right != nullptr)

make temp = it’s right node

else

{

set temp-> right to be add;

set add->parent = temp;

return;

}

}

}

}