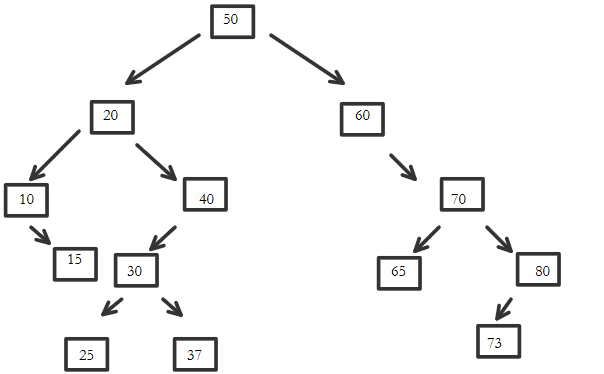
**CS32 Homework 5**

**Ben He**

**UID: 804962948**

**1(a)**



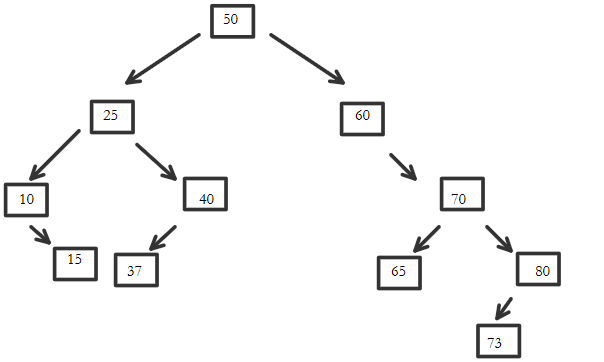
**1(b)**

In-order: 10, 15, 20, 25, 30, 37, 40, 50, 60, 65, 70, 73, 80

Pre-order: 50, 20, 10, 15, 40, 30, 25, 37, 60, 70, 65, 80, 73

Post-order: 15, 10, 25, 37, 30, 40, 20, 65, 73, 80, 70, 60, 50

**1(c)**



**2(a)**

struct Node{

int m\_data;

Node\* m\_parent;

Node\* m\_left;

Node\* m\_right;

};

**2(b)**

void insertNode(int data, Node\*& root){

if root points to null (tree is empty):

allocate new Node;

assign data to new Node;

assign null to the parent, left, and right Node pointers of new Node;

assign root to new Node;

return;

start from root of BST;

create Node pointer for current Node;

while not done:

if data equals the current Node’s value:

return;

if data is less than current Node’s value:

if there’s a left child:

move current Node to the left child;

else:

allocate new Node;

assign data to new Node;

set new Node’s left and right pointers to null;

set new Node’s parent pointer to the current Node;

set the current Node’s left pointer to new Node;

return;

if data is greater than current Node’s value:

if there’s a right child:

move current Node to right child;

else:

allocate new Node;

assign data to new Node;

set new Node’s left and right pointers to null;

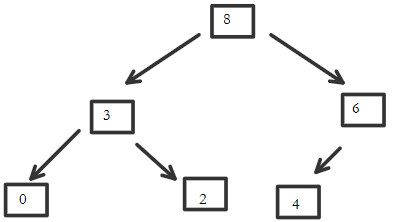
set new Node’s parent pointer to current Node;

set current Node’s right pointer to new Node;

return;

}

**3(a)**



**3(b)**

heap[] = {8, 3, 6, 0, 2, 4, ………};

**3(c)**

heap[] = {6, 3, 4, 0, 2, ………};

**4(a)** O(C + S)

**4(b)** O(log(C) + S)

**4(c)** O(log(C) + log(S))

**4(d)** O(log(S))

**4(e)** O(1)

**4(f)** O(log(C) + S)

**4(g)** O(S\*log(S))

**4(h)** O(C\*log(S))