class node {

public:

int data;

node \*left, \*right;

};

node\* newNode(int d)

{

node\* temp;

temp = new node;

temp->data = d;

temp->left = temp->right = NULL;

return temp;

}

void inorder(node\* root)

{

if (root != NULL) {

inorder(root->left);

cout << root->data<<" ";

inorder(root->right);

}

}

node\* insert(node\* node, int data)

{

if (node == NULL)

return newNode(data);

if (data < node->data)

node->left = insert(node->left, data);

else

node->right = insert(node->right, data);

return node;

}

node\* minValue(node\* node)

{

struct node \*current =node;

while (current && current->left != NULL)

current = current->left;

return current;

}

int height(node\* node)

{

if (node==NULL)

return 0;

else

{

int Rheight = height(node->right);

int Lheight = height(node->left);

if (Lheight > Rheight)

{

return(Lheight+1);

}

else

{

return(Rheight+1);

}

}

}

int TotalNodes(node\* root)

{

if (root == NULL)

return 0;

int lh = left\_height(root);

int rh = right\_height(root);

if (lh == rh)

return (1 << lh) - 1;

return 1 + TotalNodes(root->left)

+ TotalNodes(root->right);

}

Take "cur" pointer, which will point to head of the first level of the list

2) Take "tail" pointer, which will point to end of the first level of the list

3) Repeat the below procedure while "curr" is not NULL.

I) if current node has a child then

a) append this new child list to the "tail"

tail->next = cur->child

b) find the last node of new child list and update "tail"

tmp = cur->child;

while (tmp->next != NULL)

tmp = tmp->next;

tail = tmp;

II) move to the next node. i.e. cur = cur->next