1) #include <stdio.h>

#include <stdlib.h>

struct btnode

{

int value;

struct btnode \*l;

struct btnode \*r;

}\*root = NULL, \*temp = NULL, \*t2, \*t1;

/\* To create a node \*/

void create()

{

int data;

printf("Enter data of node to be inserted : ");

scanf("%d", &data);

temp = (struct btnode \*)malloc(1\*sizeof(struct btnode));

temp->value = data;

temp->l = temp->r = NULL;

}

/\* Function to search the appropriate position to insert the new node \*/

void search(struct btnode \*t)

{

if ((temp->value > t->value) && (t->r != NULL)) /\* value more than root node value insert at right \*/

search(t->r);

else if ((temp->value > t->value) && (t->r == NULL))

t->r = temp;

else if ((temp->value < t->value) && (t->l != NULL)) /\* value less than root node value insert at left \*/

search(t->l);

else if ((temp->value < t->value) && (t->l == NULL))

t->l = temp;

}

/\* To insert a node in the tree \*/

void insert()

{

create();

if (root == NULL)

root = temp;

else

search(root);

}

/\* recursive function to perform inorder traversal of tree \*/

void inorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

if (t->l != NULL)

inorder(t->l);

printf("%d -> ", t->value);

if (t->r != NULL)

inorder(t->r);

}

/\* To find the preorder traversal \*/

void preorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display");

return;

}

printf("%d -> ", t->value);

if (t->l != NULL)

preorder(t->l);

if (t->r != NULL)

preorder(t->r);

}

/\* To find the postorder traversal \*/

void postorder(struct btnode \*t)

{

if (root == NULL)

{

printf("No elements in a tree to display ");

return;

}

if (t->l != NULL)

postorder(t->l);

if (t->r != NULL)

postorder(t->r);

printf("%d -> ", t->value);

}

int main()

{

int ch;

printf("\nOPERATIONS ---");

printf("\n1 - Insert an element into tree\n");

printf("2 - Inorder Traversal\n");

printf("3 - Preorder Traversal\n");

printf("4 - Postorder Traversal\n");

printf("5 - Exit\n");

while(1)

{

printf("\nEnter your choice : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

insert();

break;

case 2:

inorder(root);

break;

case 3:

preorder(root);

break;

case 4:

postorder(root);

break;

case 5:

exit(0);

default :

printf("Wrong choice, Please enter correct choice ");

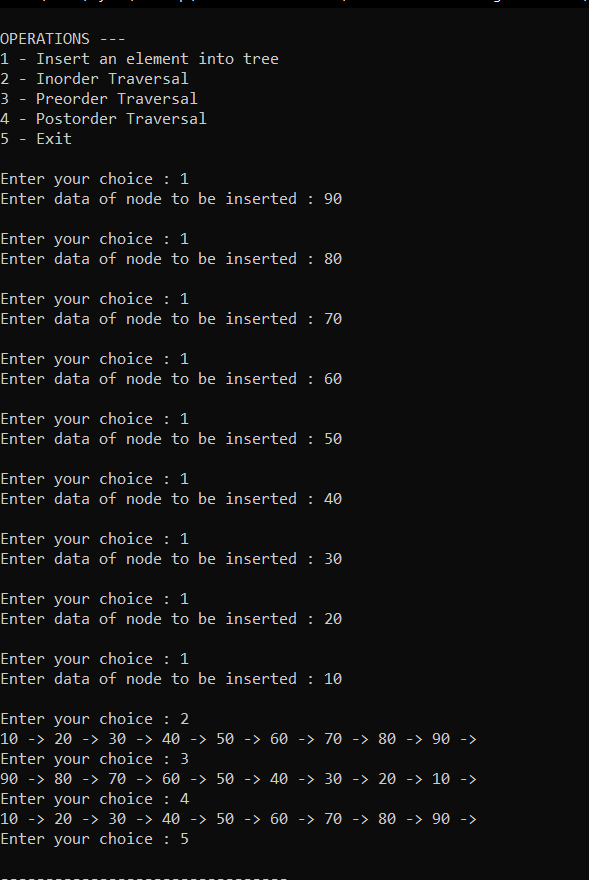
break;

}

}

return 0;

}



2) // C implementation to find leaf count of a given Binary tree

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

/\* A binary tree node has data, pointer to left child

and a pointer to right child \*/

struct node

{

int data;

struct node\* left;

struct node\* right;

};

/\* Function to get the count of leaf nodes in a binary tree\*/

int getLeafCount(struct node\* node)

{

if(node == NULL)

return 0;

if(node->left == NULL && node->right==NULL)

return 1;

else

return getLeafCount(node->left)+

getLeafCount(node->right);

}

/\* Computes the number of non-leaf nodes in a tree. \*/

int countNonleaf(struct node\* node)

{

// Base cases.

if (node == NULL || (node->left == NULL &&

node->right == NULL))

return 0;

// If root is Not NULL and its one of its

// child is also not NULL

return 1 + countNonleaf(node->left) +

countNonleaf(node->right);

}

/\* Helper function that allocates a new node with the

given data and NULL left and right pointers. \*/

struct node\* newNode(int data)

{

struct node\* node = (struct node\*)

malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

/\*Driver program to test above functions\*/

int main()

{

/\*create a tree\*/

struct node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

root->right->left=newNode(6);

root->right->right=newNode(7);

/\*get leaf count of the above created tree\*/

printf("Leaf count of the tree is %d\n", getLeafCount(root));

printf("Internal nodes of the tree is %d\n", countNonleaf(root)-1);

int sum=getLeafCount(root)+countNonleaf(root);

int height=ceil(log2(sum + 1)) - 1;

printf("Total Nodes of the tree is %d\n",sum);

printf("Height of the tree is %d\n",height);

getchar();

return 0;

}

