/\*binary search tree insert,traversal,search\*/

#include<stdlib.h>

#include<stdio.h>

struct node {

int data;

struct node\* right, \* left;

};

struct node\*tree;

void insert(struct node \*\* tree, int val)

{

struct node \*temp = NULL;

if(!(\*tree))

{

temp = (struct node \*)malloc(sizeof(struct node\*));

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

temp->data = val;

\*tree = temp;

return;

}

if(val < (\*tree)->data)

{

insert(&(\*tree)->left, val);

}

else if(val > (\*tree)->data)

{

insert(&(\*tree)->right, val);

}

}

void print\_preorder(struct node\* tree)

{

if (tree)

{

printf("%d\n",tree->data);

print\_preorder(tree->left);

print\_preorder(tree->right);

}

}

void print\_inorder(struct node \* tree)

{

if (tree)

{

print\_inorder(tree->left);

printf("%d\n",tree->data);

print\_inorder(tree->right);

}

}

void print\_postorder(struct node \* tree)

{

if (tree)

{

print\_postorder(tree->left);

print\_postorder(tree->right);

printf("%d\n",tree->data);

}

}

struct node\* search(struct node \*\* tree, int val)

{

if(!(\*tree))

{

return NULL;

}

if(val < (\*tree)->data)

{

search(&((\*tree)->left), val);

}

else if(val > (\*tree)->data)

{

search(&((\*tree)->right), val);

}

else if(val == (\*tree)->data)

{

return \*tree;

}

}

int main()

{

node \*root;

node \*tmp;

//int I;

root = NULL;

/\* Inserting nodes into tree \*/

insert(&root, 9);

insert(&root, 4);

insert(&root, 15);

insert(&root, 6);

insert(&root, 12);

insert(&root, 17);

insert(&root, 2);

/\* Printing nodes of tree \*/

printf("Pre Order Display\n");

print\_preorder(root);

printf("In Order Display\n");

print\_inorder(root);

printf("Post Order Display\n");

print\_postorder(root);

/\* Search node into tree \*/

tmp = search(&root, 4);

if (tmp)

{

printf("Searched node=%d\n", tmp->data);

}

else

{

printf("Data Not found in tree.\n");

}

}

