Tanmay Pawan Bisen Shift-2 Roll No.: B81

07 PRACTICAL SUBMISSION:

**PROBLEM DEFINITION:**

E-7: Construct a Binary Search Tree and perform the following Operations

1) In order Traversal

2) Preorder Traversal

3)Postorder Traversal

4) Count Number of nodes in Tree

5) Count Number of leaf nodes in Tree

6) Count Number of non-leaf nodes in Tree

7) Count Number of nodes with one child

**CODE:**

#include <stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*left,\*right;

};

int counter=0;

struct node \*create(struct node \*root)

{

struct node \*ptr,\*prev,\*temp;

int i,a[20],n;

printf("enter total no of nodes in tree\n");

scanf("%d",&n);

for (i=0;i<n;i++)

{

printf("Enter node %d : ",i+1);

scanf("%d",&a[i]);

}

for (i=0;i<n;i++)

{

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

temp->data=a[i];

temp->left=NULL;

temp->right=NULL;

if(root==NULL)

{

root=temp;

}

else{

ptr=root;

while(ptr!=NULL)

{

prev=ptr;

if(ptr->data < temp->data)

ptr=ptr->right;

else

ptr=ptr->left;

}

if(prev->data < temp->data)

prev->right=temp;

else

prev->left=temp;

}

}

return root;

}

void inorder\_trav(struct node \*node1)

{

if(node1==NULL)

return;

inorder\_trav(node1->left);

printf("%d",node1->data);

inorder\_trav(node1->right);

}

void preorder\_trav(struct node \*node1)

{

if(node1==NULL)

return;

printf("%d",node1->data);

preorder\_trav(node1->left);

preorder\_trav(node1->right);

}

void postorder\_trav(struct node \*node1)

{

if(node1==NULL)

return;

postorder\_trav(node1->left);

postorder\_trav(node1->right);

printf("%d",node1->data);

}

int n\_counter(struct node \*root)

{

if(root==NULL)

{

return 0;

}

return 1 + n\_counter(root->left) + n\_counter(root->right);

}

int leaf\_counter(struct node \*root)

{

if(root==NULL)

{

return 0;

}

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

{

return 1;

}

return leaf\_counter(root->left) + leaf\_counter(root->right);

}

int nonleaf\_counter(struct node \*root)

{

int k1,k2;

k1=n\_counter(root);

k2=leaf\_counter(root);

return k1-k2;

}

void child\_counter(struct node \*root)

{

if(root==NULL)

{

return;

}

if((root->left==NULL && root->right!=NULL) || (root->left!=NULL && root->right==NULL))

{

counter++;

}

child\_counter(root->left);

child\_counter(root->right);

}

int main()

{

int val;

struct node \*root;

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

root = NULL;

root=create(root);

printf("enter 0 to exit\n");

do

{

printf("\n\npress:\n 1 preorder traversal\n 2 inorder traversal\n 3 postorder traversal\n 4 number of nodes \n 5 number of leaf nodes\n 6 number of non leaf nodes\n 7 number of nodes with one child\n");

scanf("%d",&val);

switch(val)

{

case 1:

preorder\_trav(root);

break;

case 2:

inorder\_trav(root);

break;

case 3:

postorder\_trav(root);

break;

case 4:

counter=n\_counter(root);

printf("%i",counter);

break;

case 5:

counter=leaf\_counter(root);

printf("%i",counter);

break;

case 6:

counter=nonleaf\_counter(root);

printf("%i",counter);

break;

case 7:

counter=0;

child\_counter(root);

printf("%i",counter);

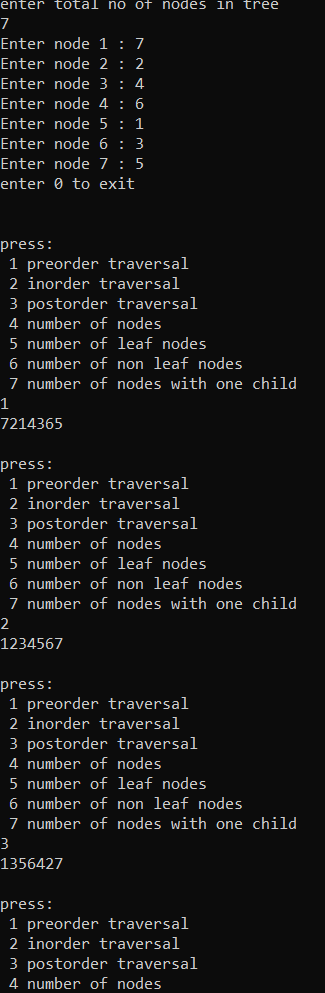
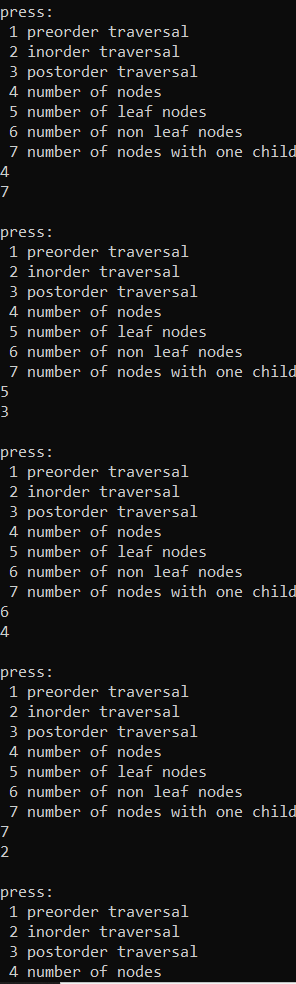
break;

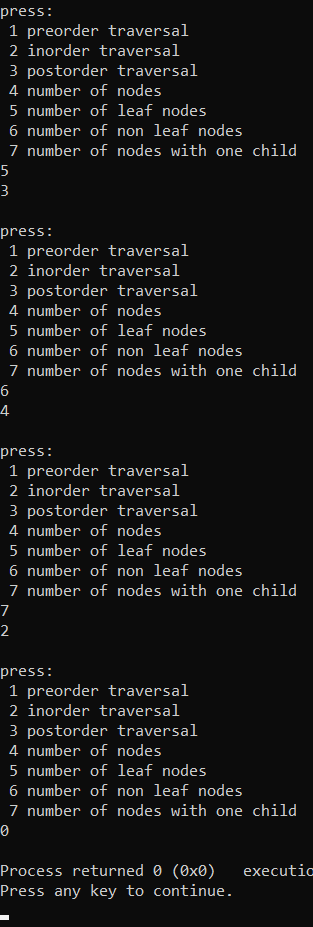
}

}while(val!=0);

return 0;

}





**---------------------------------------------------------------------------------------------------------**