Experiment 9

**Aim:-** Implement binary search tree ADT operations (Insert,Traverse)using linked list.

**Code:-**

#include "stdio.h"

#include "stdlib.h"

**struct** node

{

**int** data;

**struct** node \*left;

**struct** node \*right;

};

**struct** node \*tree;

**void** create\_tree(**struct** node \*);

**struct** node \*insertElement(**struct** node \*, **int**);

**void** preorderTraversal(**struct** node \*);

**void** inorderTraversal(**struct** node \*);

**void** postorderTraversal(**struct** node \*);

**int** totalNodes(**struct** node \*);

**int** totalExternalNodes(**struct** node \*);

**int** totalInternalNodes(**struct** node \*);

**int** Height(**struct** node \*);

**struct** node \*deleteTree(**struct** node \*);

**int** main()

{

**int** option, val;

create\_tree(tree);

printf("\n 1. Insert Element");

printf("\n 2. Preorder Traversal");

printf("\n 3. Inorder Traversal");

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

printf("\n 5. Quit");

**do**

{

printf("\n\n Enter your option : ");

scanf("%d", &option);

**switch**(option)

{

**case** 1:

printf("\n Enter the value of the new node : ");

scanf("%d", &val);

tree = insertElement(tree, val);

**break**;

**case** 2:

printf("\n The elements of the tree are : \n");

preorderTraversal(tree);

**break**;

**case** 3:

printf("\n The elements of the tree are : \n");

inorderTraversal(tree);

**break**;

**case** 4:

printf("\n The elements of the tree are : \n");

postorderTraversal(tree);

**break**;

}

}**while**(option!=5);

**return** 0;

}

**void** create\_tree(**struct** node \*tree){

tree = **NULL**;

}

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

{

**struct** node \*ptr, \*nodeptr, \*parentptr;

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

ptr->data = val;

ptr->left = **NULL**;

ptr->right = **NULL**;

**if**(tree==**NULL**){

tree=ptr;

tree->left=**NULL**;

tree->right=**NULL**;

}

**else**{

parentptr=**NULL**;

nodeptr=tree;

**while**(nodeptr!=**NULL**){

parentptr=nodeptr;

**if**(val<nodeptr->data)

nodeptr=nodeptr->left;

**else**

nodeptr = nodeptr->right;

}

**if**(val<parentptr->data)

parentptr->left = ptr;

**else**

parentptr->right = ptr;

}

**return** tree;

}

**void** preorderTraversal(**struct** node \*tree){

**if**(tree != **NULL**){

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

preorderTraversal(tree->left);

preorderTraversal(tree->right);

}

}

**void** inorderTraversal(**struct** node \*tree){

**if**(tree != **NULL**){

inorderTraversal(tree->left);

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

inorderTraversal(tree->right);

}

}

**void** postorderTraversal(**struct** node \*tree){

**if**(tree != **NULL**){

postorderTraversal(tree->left);

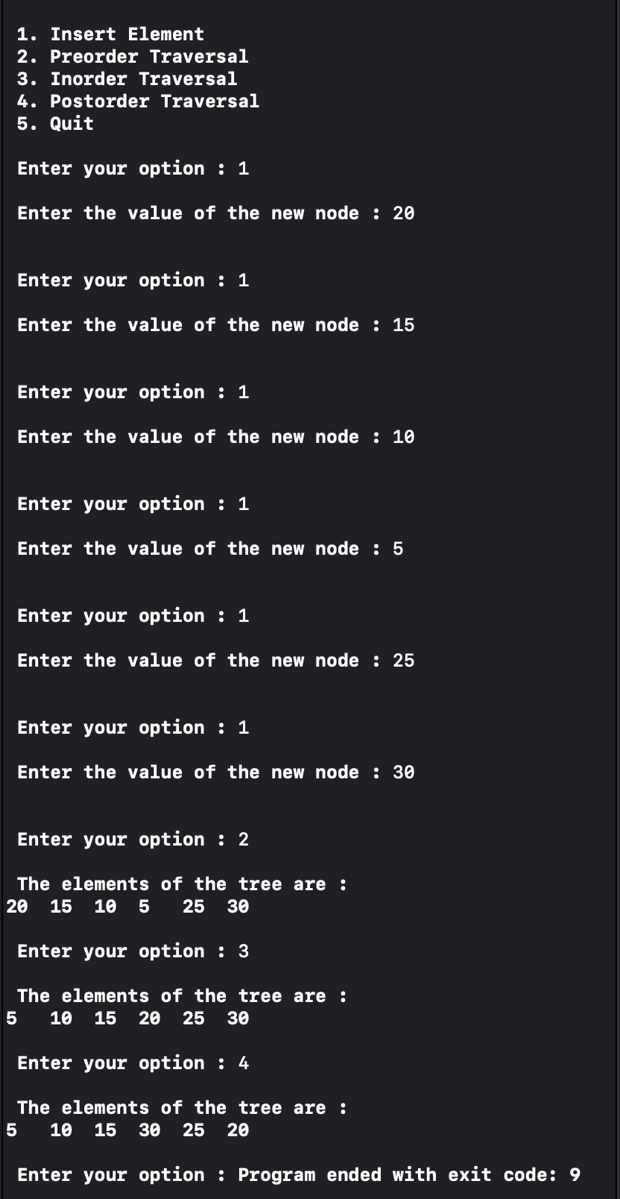
postorderTraversal(tree->right);

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

}

}

**Output :-**



**Conclusion:-** Thus, we have implement binary search tree ADT operations (Insert,Traverse)using linked list.