AIM:-

Aassignment 11

Create a binary search tree( BST)

a)Perform recursive inorder , preorder and postorder traversals.

b) For BST, Perform nonrecursive inorder , preorder and postorder traversal.

OBJECTIVE:-

Traversing a tree means visiting every node in the tree. You might for instance want to add all the values in the tree or find the largest one. For all these operations, you will need to visit each node of the tree.

Theory :-

1. Uses of Inorder In case of binary search trees (BST), Inorder traversal gives nodes in non- decreasing order. To get nodes of BST in non-increasing order, a variation of Inorder traversal where Inorder traversal s reversed can be used.
2. Preorder traversal is used to create a copy of the tree. Preorder traversal is also used to get prefix expression on of an expression tree.
3. Postorder traversal is used to delete the tree.To delete a tree we must traverse all the nodes of the tree and delete them one by one,because before deleting the parent node we should delete its children nodes first.

Postorder traversal is also useful to get the postfix expression of an expression tree.

Sourcecode :-

#include<iostream>

using namespace std;

struct node

{

int key;

struct node \*left, \*right;

};

struct node \*newNode(int item)

{

struct node \*temp;

temp=new node;

temp->key = item;

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

return temp;

}

void printPostorder(struct node \*node)

{

if (node == NULL)

{

return;

}

printPostorder(node->left);

printPostorder(node->right);

cout<< node->key << " ";

}

void printInorder(struct node \*node)

{

if (node == NULL)

{

return;

}

printInorder(node->left);

cout << node->key << " ";

printInorder(node->right);

}

void printPreorder(struct node \*node)

{

if (node == NULL)

{

return;

}

cout << node->key << " ";

printPreorder(node->left);

printPreorder(node->right);

}

struct node\* insert(struct node\* node, int key)

{

if (node == NULL)

{

return newNode(key);

}

if (key < node->key)

{

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

}

else if (key > node->key)

{

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

}

return node;

}

int main()

{

struct node \*root = NULL;

int j,i,n;

cout<<"Enter number of elements :";

cin>>n;

cout<<"Enter elements:"<<endl;

cin>>j;

root = insert(root, j);

for(i=1;i<n;i++)

{

cin>>j;

insert(root,j);

}

cout << "\nPreorder traversal of binary tree is \n";

printPreorder(root);

cout << "\nInorder traversal of binary tree is \n";

printInorder(root);

cout << "\nPostorder traversal of binary tree is \n";

printPostorder(root);

return 0;

}

Output:-

/\*

Enter number of elements :7

Enter elements:

3

4

2

1

6

5

7

Preorder traversal of binary tree is

3 2 1 4 6 5 7

Inorder traversal of binary tree is

1 2 3 4 5 6 7

Postorder traversal of binary tree is

1 2 5 7 6 4 3

--------------------------------

Process exited after 8.729 seconds with return value 0

Press any key to continue . . .

\*/

Conclusion:-

Created a binary search tree( BST)

a)Perform recursive inorder , preorder and postorder traversals.

b) For BST, Perform nonrecursive inorder , preorder and postorder traversal.