Practical No:- 7

1. write a program to create the tree and display the elements.
2. Write a program to construct the binary tree.
3. Write a program to construct the binary tree. Write a program for inorder , postorder and preorder traversal of tree code.

Practical Implementation:-

Code:-

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

and a pointer to right child \*/

struct Node {

char data;

struct Node\* left;

struct Node\* right;

};

struct Node\* newNode(char data)

{

struct Node\* node = new Node;

node->data = data;

node->left = node->right = NULL;

return (node);

}

/\* Recursive function to construct binary of size

len from Inorder traversal in[] and Preorder traversal

pre[]. Initial values of inStrt and inEnd should be

0 and len -1. The function doesn't do any error

checking for cases where inorder and preorder

do not form a tree \*/

struct Node\* buildTree(char in[], char pre[], int inStrt,

int inEnd, unordered\_map<char, int>& mp)

{

static int preIndex = 0;

if (inStrt > inEnd)

return NULL;

/\* Pick current node from Preorder traversal using preIndex

and increment preIndex \*/

char curr = pre[preIndex++];

struct Node\* tNode = newNode(curr);

/\* If this node has no children then return \*/

if (inStrt == inEnd)

return tNode;

/\* Else find the index of this node in Inorder traversal \*/

int inIndex = mp[curr];

/\* Using index in Inorder traversal, construct left and

right subtress \*/

tNode->left = buildTree(in, pre, inStrt, inIndex - 1, mp);

tNode->right = buildTree(in, pre, inIndex + 1, inEnd, mp);

return tNode;

}

// This function mainly creates an unordered\_map, then

// calls buildTree()

struct Node\* buldTreeWrap(char in[], char pre[], int len)

{

// Store indexes of all items so that we

// we can quickly find later

unordered\_map<char, int> mp;

for (int i = 0; i < len; i++)

mp[in[i]] = i;

return buildTree(in, pre, 0, len - 1, mp);

}

/\* This function is here just to test buildTree() \*/

void printInorder(struct Node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

printf("%c ", node->data);

printInorder(node->right);

}

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

int main()

{

char in[] = { 'D', 'B', 'E', 'A', 'F', 'C' };

char pre[] = { 'A', 'B', 'D', 'E', 'C', 'F' };

int len = sizeof(in) / sizeof(in[0]);

struct Node\* root = buldTreeWrap(in, pre, len);

Inorder traversal \*/

printf("Inorder traversal of the constructed tree is \n");

printInorder(root);

}

Output:-

