**Topic – Creating Binary Tree with Given Preorder and Inorder String of Characters**

* **Problem Statement**

**Write a program to create a binary tree given its preorder and inorder traversal as a string of chars. Then write a function to display the tree by traversing inorder.**

**Input and Output example:**

**The preorder input of characters are: DBJAECFI**

**The inorder input of characters are: DBEAFICJ**

**The inorder traversal of the tree built by the given preoder and inorder input is : D B E A F I C J**

* **Proposed C Code**

**/\* ---------- preorder\_inorder.c--------------- \*/**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**/\* Tree of characters \*/**

**typedef struct node**

**{**

**char data;**

**struct node\* lchild;**

**struct node\* rchild;**

**}Tree;**

**/\* creating node \*/**

**Tree\* creation(char data)**

**{**

**Tree\* ptr = (Tree\*)malloc(sizeof(Tree));**

**ptr->data = data;**

**ptr->lchild = NULL;**

**ptr->rchild = NULL;**

**return ptr;**

**}**

**/\* searching for the position of a charcter \*/**

**int search(char\* inorder,char currentnode,int start,int end)**

**{**

**for ( int i = start ; i <= end ; i++ )**

**{**

**if ( inorder[i] == currentnode )**

**{**

**return i;**

**}**

**}**

**return -1;**

**}**

**/\* building the tree \*/**

**static int preindex = 0;**

**Tree\* building\_of\_Tree(char\* preorder,char\* inorder,int start,int end)**

**{**

**if ( start > end )**

**{**

**return NULL;**

**}**

**char currentnode = preorder[preindex++];/\* current data \*/**

**Tree\* root = creation(currentnode);**

**if ( start == end )**

**{**

**return root;**

**}**

**int root\_position = search(inorder,currentnode,start,end);**

**root->lchild = building\_of\_Tree(preorder,inorder,start,root\_position-1);/\* creating left subtree \*/**

**root->rchild = building\_of\_Tree(preorder,inorder,root\_position+1,end);/\* creating right subtree \*/**

**return root;**

**}**

**/\* inorder traversal to print the nodes of binary tree \*/**

**void inorderTraversal(Tree\* root)**

**{**

**if(root == NULL)**

**{**

**return;**

**}**

**inorderTraversal(root->lchild);/\* traversing for left child \*/**

**printf("%c ",root->data);/\* printing data inorder \*/**

**inorderTraversal(root->rchild);/\* traversing for right child \*/**

**}**

**int main()**

**{**

**char preorder[] = {'D', 'B', 'J' , 'A', 'E', 'C', 'F' , 'I'};/\* preorder input \*/**

**char inorder[] = {'D', 'B', 'E', 'A', 'F', 'I' , 'C' , 'J'};/\* inorder input \*/**

**int length = sizeof(inorder) / sizeof(inorder[0]);**

**printf("The preorder input of characters are: ");**

**for ( int i = 0 ; i < length ; i++ )**

**{**

**printf("%c",preorder[i]);**

**}**

**printf("\n");**

**printf("The inorder input of characters are: ");**

**for ( int i = 0 ; i < length ; i++ )**

**{**

**printf("%c",inorder[i]);**

**}**

**printf("\n");**

**Tree\* root = building\_of\_Tree(preorder,inorder,0,length-1);**

**printf("The inorder traversal of the tree built by the given preoder and inorder input is : ");**

**inorderTraversal(root);/\* inorder traversal \*/**

**return 0;**

**}**

**/\*------------------------------------------------------------------------------------------------------------------------- \*/**

* **Conclusion**

**The proposed algorithm has average time complexity of O(n^2) where n is the input size i.e the size of input preorder and inorder charcter array.**

* **Limitations : The input preorder and inorder character array should contain same characters but should be in correct order.**
* **Assumptions: The input i.e preorder and inoder array contains charcters and it is taken in the main program .**