**Binary Tree**

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 example :

1. 25->12->10->17->27->26->28
2. 10->12->17->25->26->27->28

* Output example :

1. 10 ->12 ->17 ->25 ->26 ->27 ->28 ->
2. 10 ->12 ->17 ->25 ->26 ->27 ->28 ->

Proposed C Code:

/\* ------- main.c ------- \*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

// Creating the tree

typedef struct N

{

int val;

struct N \*right;

struct N \*left;

} Node;

// Initializing the tree

Node \*root = NULL;

// Creating a new node

Node \*createNode(int val)

{

Node \*branch = (Node \*)malloc(sizeof(Node));

branch->val = val;

branch->left = NULL;

branch->right = NULL;

return branch;

}

// Inserting a node

Node \*insertNode(Node \*root, int val)

{

if (root == NULL)

{

return createNode(val);

}

if (val > root->val)

{

root->right = insertNode(root->right, val);

}

else

{

root->left = insertNode(root->left, val);

}

return root;

}

// Entering values in the tree from the input

void enter(char \*s)

{

int curr = 0;

for (int i = 0; i < strlen(s); i++)

{

if ((s[i] - '0') >= 0 && (s[i] - '0') <= 9)

{

curr = curr \* 10 + (s[i] - '0');

}

else if (s[i] == '-' && s[i+1]=='>')

{

root = insertNode(root, curr);

curr = 0;

}

}

root = insertNode(root, curr);

}

// Display the tree

void display(Node \*root)

{

if (root == NULL)

{

return;

}

display(root->left);

printf("%d ->", root->val);

display(root->right);

}

int main()

{

char \*s = (char \*)malloc(100000 \* sizeof(char));

printf("Enter the input in num->num->num->num format(Do not use spaces) in Inorder/preorder manner:\n");

scanf("%[^\n]s", s);

enter(s);

printf("The inorder form is: ");

display(root);

return 0;

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

Conclusion:

The proposed algorithm has a runtime of O(n), where n is the size of the input string.

Limitations and assumptions for this algorithm include:

1.Here insertion and deletion are take place at O(n) time in stead of constant time(O(1)).

2.Here maximum input string size is 100000.

3.In the input string do not use any spaces. Enter the input string in “n->n->n->n” format.