Program :-

#include <stdio.h>

#include <stdlib.h>

struct Node

{

    int data;

    struct Node \*left, \*right;

    int leftThread;

    int rightThread;

};

struct Node \*insert(struct Node \*root, int key)

{

    struct Node \*ptr = root;

    struct Node \*parent = NULL;

    while (ptr != NULL)

    {

        if (key == (ptr->data))

        {

            printf("Duplicate Key !\n");

            return root;

        }

        parent = ptr;

        if (key < ptr->data)

        {

            if (ptr->leftThread == 0)

                ptr = ptr->left;

            else

                break;

        }

        else

        {

            if (ptr->rightThread == 0)

                ptr = ptr->right;

            else

                break;

        }

    }

    struct Node \*value = (struct Node \*)malloc(sizeof(struct Node));

    value->data = key;

    value->leftThread = 1;

    value->rightThread = 1;

    if (parent == NULL)

    {

        root = value;

        value->left = NULL;

        value->right = NULL;

    }

    else if (key < (parent->data))

    {

        value->left = parent->left;

        value->right = parent;

        parent->leftThread = 0;

        parent->left = value;

    }

    else

    {

        value->left = parent;

        value->right = parent->right;

        parent->rightThread = 0;

        parent->right = value;

    }

    return root;

}

struct Node \*inorderSuccessor(struct Node \*ptr)

{

    if (ptr->rightThread == 1)

        return ptr->right;

    ptr = ptr->right;

    while (ptr->leftThread == 0)

        ptr = ptr->left;

    return ptr;

}

void inorder(struct Node \*root)

{

    if (root == NULL)

        printf("Tree is empty");

    struct Node \*ptr = root;

    while (ptr->leftThread == 0)

        ptr = ptr->left;

    while (ptr != NULL)

    {

        printf("%d ", ptr->data);

        ptr = inorderSuccessor(ptr);

    }

}

int main()

{

    struct Node \*root = NULL;

    int key;

    char choice;

    do

    {

        printf("Enter the key to insert: ");

        scanf("%d", &key);

        root = insert(root, key);

        printf("Do you want to insert another key? (y/n): ");

        scanf(" %c", &choice);

    } while (choice == 'y' || choice == 'Y');

    printf("Inorder traversal of the threaded binary tree: ");

    inorder(root);

    return 0;

}

Output :-

PS C:\Users\HP\Desktop\coding\C> cd "c:\Users\HP\Desktop\coding\C\" ; if ($?) { gcc tbt.c -o tbt } ; if ($?) { .\tbt }

Enter the key to insert: 23

Do you want to insert another key? (y/n): y

Enter the key to insert: 4

Do you want to insert another key? (y/n): y

Enter the key to insert: 30

Do you want to insert another key? (y/n): y

Enter the key to insert: 11

Do you want to insert another key? (y/n): y

Enter the key to insert: 7

Do you want to insert another key? (y/n): y

Enter the key to insert: 34

Do you want to insert another key? (y/n): y

Enter the key to insert: 20

Do you want to insert another key? (y/n): y

Enter the key to insert: 24

Do you want to insert another key? (y/n): y

Enter the key to insert: 1

Do you want to insert another key? (y/n): n

Inorder traversal of the threaded binary tree: 1 4 7 11 20 23 24 30 34

PS C:\Users\HP\Desktop\coding\C>