**PROBLEM STATEMENT:**

Write a program to form a lexical (ordered) tree. Include member functions to check if the tree so formed is a complete binary tree.

**PROGRAM CODE:**

**#include<iostream>**

**#include<conio.h>**

**using namespace std;**

**struct node**

**{**

**int info;**

**node\* left;**

**node\* right;**

**};**

**class tree //Binary Search Tree (Ordered lexically or numerically)**

**{**

**private:**

**node\* root;**

**node\* ptr;**

**node\* par;**

**node\* stack[50];**

**public:**

**tree()**

**{**

**root=NULL;**

**}**

**node\* newnode(int x)**

**{**

**node\* temp;**

**temp=new node;**

**temp->right=NULL;**

**temp->left=NULL;**

**temp->info=x;**

**return temp;**

**}**

**void insert(int a)**

**{**

**node\* n=newnode(a);**

**ptr=root;**

**par=NULL;**

**while(ptr!=NULL)**

**{**

**if(ptr->info==a)**

**{**

**cout<<"\nItem already exists!!\n";**

**return;**

**}**

**par=ptr;**

**if(a<ptr->info)**

**ptr=ptr->left;**

**else**

**ptr=ptr->right;**

**}**

**if(par==NULL)**

**{**

**root=n;**

**return;**

**}**

**else**

**{**

**if(a<par->info)**

**par->left=n;**

**else**

**par->right=n;**

**}**

**return;**

**}**

**void disp()**

**{**

**cout<<"\nBinary Tree (In-order): ";**

**int top=0;**

**stack[top]=NULL;**

**ptr=root;**

**y:**

**while(ptr!=NULL)**

**{**

**top++;**

**stack[top]=ptr;**

**ptr=ptr->left;**

**}**

**ptr=stack[top];**

**top--;**

**while(ptr!=NULL)**

**{**

**cout<<ptr->info<<" ";**

**if(ptr->right!=NULL)**

**{**

**ptr=ptr->right;**

**goto y;**

**}**

**ptr=stack[top];**

**top--;**

**}**

**return;**

**}**

**bool check()**

**{**

**int top=0;**

**stack[top]=NULL;**

**ptr=root;**

**y:**

**while(ptr!=NULL)**

**{**

**top++;**

**stack[top]=ptr;**

**ptr=ptr->left;**

**}**

**ptr=stack[top];**

**top--;**

**while(ptr!=NULL)**

**{**

**if(((ptr->right==NULL)&&(ptr->left!=NULL))||((ptr->right!=NULL)&&(ptr->left==NULL)))**

**return 0;**

**if(ptr->right!=NULL)**

**{**

**ptr=ptr->right;**

**goto y;**

**}**

**ptr=stack[top];**

**top--;**

**}**

**return 1;**

**}**

**};**

**int main()**

**{**

**int e,s;**

**bool chk;**

**char ch;**

**tree t1;**

**cout<<"\n\*\*\*\*\*\*\*\*\*\* Lets create a binary search tree!! \*\*\*\*\*\*\*\*\*\*\n";**

**z:**

**cout<<"\nChoose:\n1. Enter element to the tree\n2. Check if it is a complete binary tree\n3. Exit\n";**

**cin>>s;**

**switch(s)**

**{**

**case 1:**

**ch='y';**

**while(ch=='y')**

**{**

**cout<<"\nEnter the element: ";**

**cin>>e;**

**t1.insert(e);**

**t1.disp();**

**cout<<" continue? (y/n): ";**

**cin>>ch;**

**}**

**goto z;**

**case 2:**

**chk=t1.check();**

**if(chk)**

**cout<<"\nIt is a complete binary tree!!\n";**

**else**

**cout<<"\nIt is not a complete binary tree!!\n";**

**getch();**

**goto z;**

**case 3:**

**break;**

**}**

**return 0;**

**}**

**OUTPUT:**

\*\*\*\*\*\*\*\*\*\* Lets create a binary search tree!! \*\*\*\*\*\*\*\*\*\*

Choose:

1. Enter element to the tree

2. Check if it is a complete binary tree

3. Exit

1

Enter the element: 5

Binary Tree (In-order): 5 continue? (y/n): y

Enter the element: 1

Binary Tree (In-order): 1 5 continue? (y/n): y

Enter the element: 3

Binary Tree (In-order): 1 3 5 continue? (y/n): y

Enter the element: 7

Binary Tree (In-order): 1 3 5 7 continue? (y/n): n

Choose:

1. Enter element to the tree

2. Check if it is a complete binary tree

3. Exit

2

It is not a complete binary tree!!

Choose:

1. Enter element to the tree

2. Check if it is a complete binary tree

3. Exit

1

Enter the element: -1

Binary Tree (In-order): -1 1 3 5 7 continue? (y/n): n

Choose:

1. Enter element to the tree

2. Check if it is a complete binary tree

3. Exit

2

It is a complete binary tree!!

Choose:

1. Enter element to the tree

2. Check if it is a complete binary tree

3. Exit

1

Enter the element: 8

Binary Tree (In-order): -1 1 3 5 7 8 continue? (y/n): y

Enter the element: 6

Binary Tree (In-order): -1 1 3 5 6 7 8 continue? (y/n): n

Choose:

1. Enter element to the tree

2. Check if it is a complete binary tree

3. Exit

2

It is a complete binary tree!!

**RESULT:**

Hence a binary search tree is created. Member functions are given to check, at any point in the program, if the formed tree is complete binary tree.