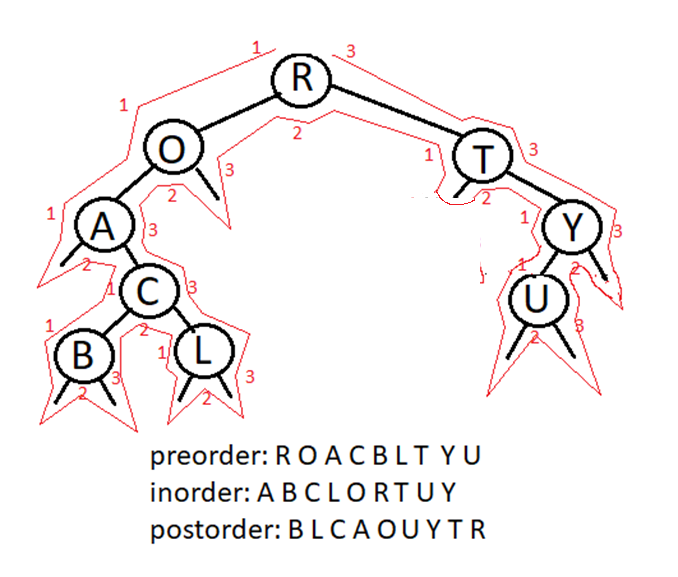
Kahlil Bello

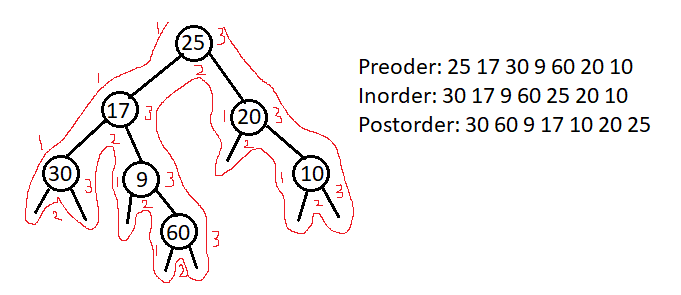
Project 9

11/3/2018

**Project 9 Problem 1---------------------------------------------------------------------------------------------------------**

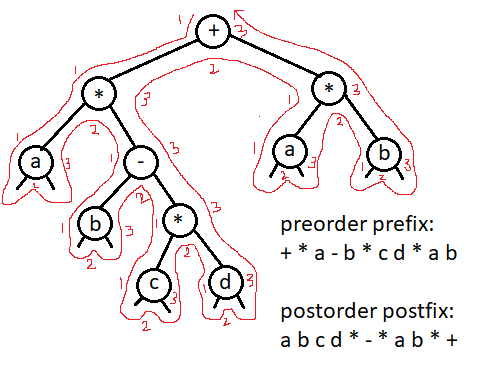


**Project 9 prob 2---------------------------------------------------------------------------------------------------------**



**Project 9 prob 3---------------------------------------------------------------------------------------------------------**

//construct tree and apply postfix and prefix



**Project 9 Prob 4---------------------------------------------------------------------------------------------------------**

HEADER FILE

#pragma once

#ifndef BST

#define BST

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

class BTREE

{

private:

struct node

{

string day;

node \*left;

node \*right;

};//end node

node \*root; //we want root to point to new node

public:

//set root to null and we use constructor

BTREE() { root = NULL; }

//insert x in BTREE

void insert(node \* &r, string x)

{

//create new node

node \*p= new(node);

p->day = x;

p->left = NULL;

p->right = NULL;

if (r == NULL) { r = p; }//endif

else

{

if (x < r->day) { insert(r->left, x); }

if (x > r->day) { insert(r->right, x); }

}//endelse

}//end insert

//display tree using inorder traversal

void displayInorder(node \* &r)

{

//nodes seen 2nd time (LC)(P)(RC)

if (r != NULL)

{

displayInorder(r->left); //LC

cout << r->day << " "; //P

displayInorder(r->right); //RC

}//endif

}//end displayInorder

void displayPreorder(node \* &r)

{

//nodes seen 2nd time (P)(LC)(RC)

if (r != NULL)

{

cout << r->day << " "; //P

displayInorder(r->left); //LC

displayInorder(r->right); //RC

}//endif

}//end displayPreorder

void displayPostorder(node \* &r)

{

//nodes seen 2nd time (P)(LC)(RC)

if (r != NULL)

{

displayInorder(r->left); //LC

displayInorder(r->right); //RC

cout << r->day << " "; //P

}//endif

}//end displayPostorder

void insert(string x)

{

insert(root, x); //pass root and x

}//end overriden insert

void displayTreeInorder() { displayInorder(root); }//end displayTree

void displayTreePreorder() { displayPreorder(root); }//end displayTree

void displayTreePostorder() { displayPostorder(root); }//end displayTree

void search(node \* &r, string x)

{

if (r == NULL) //if we look for something we don't find it points to NULL

{

cout << x << " Does not exist\n"; return;

}//endif

else if (x == r->day)

{

cout << x << " is found\n"; return;

}//endelseif

else

{

if (x < r->day) { search(r->left, x); }

if (x > r->day) { search(r->right, x); }

}//endelse

}//end search

void search(string x) { search(root, x); }//end overriden search

int treeHeight(node \* &r)

{

if (r == NULL) return -1;

else return 1 + max(treeHeight(r->left), treeHeight(r->right));

}//end treeHeight

int treeHeight() { return treeHeight(root); }//end overriden treeHeight

int countNodes(node \* &r)

{

if (r == NULL) return 0;

else { return 1 + countNodes(r->left) + countNodes(r->right); }

}//end countNodes

int countNodes() { return countNodes(root); }//end overriden countNodes

};//end class BTREE

#endif

//-------------------------------------------------------------------------------------------------------

SOURCE:

#include "BST.h"

using namespace std;

int main()

{

//we need a pointer to point to the tree

BTREE t; //object

//we want to insert name of months

string months[12] = { "Jan","Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" };

for (int i = 0; i < 12; ++i)

{

t.insert(months[i]); //calls the overriden insert function

}//endfor

/\*display the ree, we can't call main function bec

root is needed and is in private. Pass it to displayTree instead\*/

cout << "INORDER TRAVERSAL: ";

t.displayTreeInorder();

cout << endl;

cout << "PREORDER TRAVERSAL: ";

t.displayTreePreorder();

cout << endl;

cout << "POSTORDER TRAVERSAL: ";

t.displayTreePostorder();

cout << endl;

cout << "--------------------------------------------------------------------------\n";

char again = 'y';

string search;

do {

cout << "\nEnter something to search: ";

cin >> search;

t.search(search);

cout << "Search again? (y/n)? ";

cin >> again;

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

cout << "\n--------------------------------------------------------------------------";

cout << "\nYour tree height is: " << t.treeHeight() << endl;

cout << "\n--------------------------------------------------------------------------";

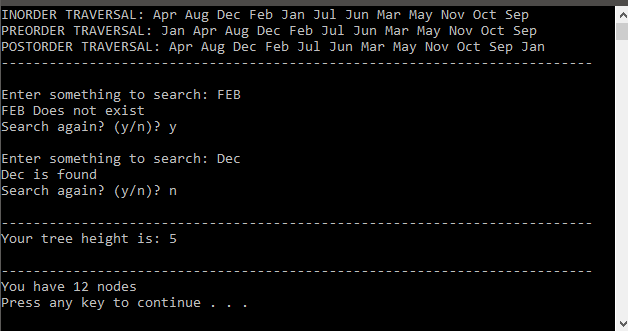
cout << "\nYou have " << t.countNodes() << " nodes" << endl;

system("pause");

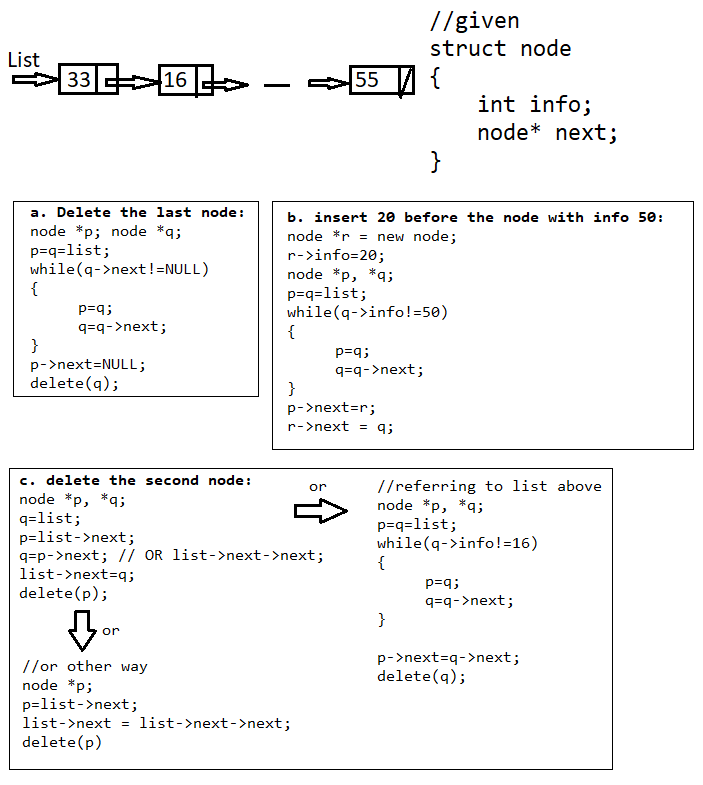
return 0;

}

OUTPUT:



**Project 9 Prob 5---------------------------------------------------------------------------------------------------------**

****

**Project 9 Prob 6---------------------------------------------------------------------------------------------------------**

Big Oh estimates

1. void f(int n)

{

if(n<=2) return 2;

else return f(n-1)+f(n-2)

}

**ANSWER: O(n^2)**

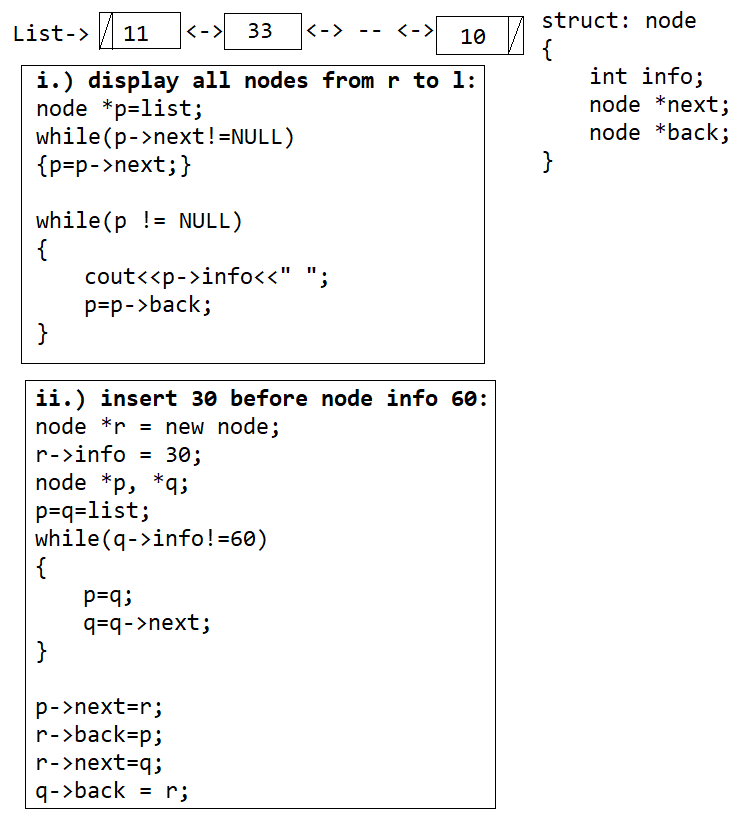
1. F(n) = (n+2)^4 (n-1)^3 + (n^2 +3) (n^4 -1)

O(n^7) + O(n^6)

max((O(n^7)) , (O(n^6)))

**ANSWER: O(n^7)**

**Project 9 Prob 7---------------------------------------------------------------------------------------------------------**

****

**Project 9 Prob 8---------------------------------------------------------------------------------------------------------**

int f(int n)

{

if(n==2)return 9;

else return f(n-1)+pow(3,n)

}