**CD Open ended problem (OEP) :**

**Write a program to generate a parse tree.**

#include <iostream>

#include <cstdlib>

#include <stack>

#include <sstream>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

class BinaryTree {

private:

string key;

BinaryTree \*leftChild;

BinaryTree \*rightChild;

public:

BinaryTree(string rootObj){

this->key = rootObj;

this->leftChild = NULL;

this->rightChild = NULL;

}

void insertLeft(string newNode){

if (this->leftChild == NULL){

this->leftChild = new BinaryTree(newNode);

}

else {

BinaryTree \*t = new BinaryTree(newNode);

t->leftChild = this->leftChild;

this->leftChild = t;

}

}

void insertRight(string newNode){

if (this->rightChild == NULL){

this->rightChild = new BinaryTree(newNode);

}

else {

BinaryTree \*t = new BinaryTree(newNode);

t->rightChild = this->rightChild;

this->rightChild = t;

}

}

BinaryTree \*getRightChild(){

return this->rightChild;

}

BinaryTree \*getLeftChild(){

return this->leftChild;

}

void setRootVal(string obj){

this->key = obj;

}

string getRootVal(){

return this->key;

}

};

BinaryTree \*buildParseTree(string fpexp){

string buf;

stringstream ss(fpexp);

vector<string> fplist;

while (ss >> buf){

fplist.push\_back(buf);

}

stack<BinaryTree\*> pStack;

BinaryTree \*eTree = new BinaryTree("");

pStack.push(eTree);

BinaryTree \*currentTree = eTree;

string arr[] = {"+", "-", "\*", "/"};

vector<string> vect(arr,arr+(sizeof(arr)/ sizeof(arr[0])));

string arr2[] = {"+", "-", "\*", "/", ")"};

vector<string> vect2(arr2,arr2+(sizeof(arr2)/ sizeof(arr2[0])));

for (unsigned int i = 0; i<fplist.size(); i++){

if (fplist[i] == "("){

currentTree->insertLeft("");

pStack.push(currentTree);

currentTree = currentTree->getLeftChild();

}

else if (find(vect.begin(), vect.end(), fplist[i]) != vect.end()){

currentTree->setRootVal(fplist[i]);

currentTree->insertRight("");

pStack.push(currentTree);

currentTree = currentTree->getRightChild();

}

else if (fplist[i] == ")"){

currentTree = pStack.top();

pStack.pop();

}

else if (find(vect2.begin(), vect2.end(), fplist[i]) == vect2.end()) {

try {

currentTree->setRootVal(fplist[i]);

BinaryTree \*parent = pStack.top();

pStack.pop();

currentTree = parent;

}

catch (string ValueError ){

cerr <<"token " << fplist[i] << " is not a valid integer"<<endl;

}

}

}

return eTree;

}

void postorder(BinaryTree \*tree){

if (tree != NULL){

postorder(tree->getLeftChild());

postorder(tree->getRightChild());

cout << tree->getRootVal() << endl;

}

}

int main() {

BinaryTree \*pt = buildParseTree("( ( 10 + 5 ) \* 3 )");

postorder(pt);

return 0;

} **Program output:**

