```
#include <iostream>
#include <string>
using namespace std;

#include "tstack.h"
#include "tqueue.h"

struct Tree
{
    string info;
    Tree * left;
    Tree * right;
    Tree(const string& s, Tree* llink, Tree * rlink)
        : info(s), left(llink), right(rlink)
    { }
};

void inorder(Tree * t)
{
    if (t != 0) {
        inorder(t->left);
        cout << t->info << endl;
        inorder(t->right);
    }
}

void preorder(Tree * t)
{
    if (t != 0) {
        cout << t->info << endl;
        preorder(t->left);
        preorder(t->right);
    }
}

void levelorder(Tree * t)
{
    tqueue<Tree *> q;
    if (t != 0) {
        q.enqueue(t);
    }
    while (! q.isEmpty()) {
        t = q.front();
        q.dequeue();
        if (t->left != 0)  q.enqueue(t->left);
        if (t->right != 0) q.enqueue(t->right);
        cout << t->info << endl;
    }
}

void whatorder(Tree * t)
{
    tstack<Tree *> s;
    while (t != 0 || ! s.isEmpty() ) {
       while (t != NULL) {
           s.push(t);
           t = t->left;
       }
       // assert: t == 0 && ! s.isEmpty()

       s.pop(t);
       cout << t->info << endl;
       t = t->right;
   }
}

void otherorder(Tree * t)
{
    tstack<Tree *> s;
```



```
    if (t != 0) {
        s.push(t);
    }
    while (! s.isEmpty()) {
        t = s.top();
        s.pop();
        if (t->right != 0) s.push(t->right);
        if (t->left != 0)  s.push(t->left);
        cout << t->info << endl;
    }
}


int main()
{
    Tree * root = new Tree("llama",
                    new Tree("elephant",
                      new Tree("cougar",
                        0,
                        new Tree("dog",0,0)),
                      new Tree("koala",0,0)),
                    new Tree("tiger",
                      new Tree("monkey",0,0),
                      new Tree("yak",
                        new Tree("warthog",0,0),
                        new Tree("zebra",0,0))));

    cout << "inorder" << endl << "-----" << endl;
    inorder(root);
    cout << endl << "preorder" << endl << "-----" << endl;
    preorder(root);
    cout << endl << "levelorder" << endl << "-----" << endl;
    levelorder(root);

    //    cout << endl << "whatorder" << endl << "-----" << endl;
    //whatorder(root);

    //cout << endl << "otherorder" << endl << "-----" << endl;
    //otherorder(root);

    return 0;

}
```

```
// minimal comments, postfix.cc, Owen Astrachan, 9/3/93
// modified 9/16/94, modified 1/27/96 for stack/class changes
// modified 9/19/97
// modified 2/00

#include <iostream>
#include <cctype>                    // for isspace

#include "tstack.h"

int Operate(char op,int left, int right);  // implemented below

int main()
{
    tstack<int> s;
    int left,right, result;
    char c;

    while (cin.get(c))              // one char at a time
    {
        if (isdigit(c))
        {
            cin.putback(c);         // back onto stream for reading
            cin >> left;            // read number, push it
            s.push(left);
        }
        else if (! isspace(c))      // operate char: '*', etc., push
        {
            s.pop(right);
            s.pop(left);
            s.push(Operate(c,left,right));
        }
        else if (c == '\n')
        {
            break;
        }
    }
    // get result (on top of stack)
    if (s.isEmpty())
    {
        cerr << "malformed postfix, stack empty" << endl;
        exit(1);
    }

    s.pop(result);                  // final answer
    if (!s.isEmpty())
    {
        cerr << "malformed postfix, stack not empty" << endl;
    }
    else
    {
        cout << "result = " << result << endl;
    }
}

int Operate(char c, int left, int right)
// precondition: c in ['+','*','-','/'], represents corresponding op
// postcondition: returns left op right
{
    int result;
    switch(c)
    {
      case '+':
        result = left+right;
        break;
      case '*':
        result =  left*right;
        break;
      case '-':
        result =  left-right;
        break;
      case '/':
        result = left/right;
        break;
      default:
        result = 0;
        cerr << "error, unknown symbol: " << c << endl;
        break;
    }
    return result;
}
```