```
// Class: CSE 330
// Term: Spring 2014
// Instructor: George M. Georgiou
// Name: Seth Lemanek
// Lab 2
// Title: Infix to postfix expression conversion
#include<iostream>
#include<stack>
#include<ctype.h>
#include<stdio.h>
#include<string>
#include<cstring>

using namespace std;
int prec(char & c)
{
   if(c == '*'||c == '/')
      return 2;
   else if(c == '+'||c == '-')
      return 1;
   else
      return 0;
}


int main()
{
 cout << "Type in an equation.\n";
 stack<char> s;
 string l;
 char * line = new char[l.size()+1];
 cin >> l;
 strcpy(line, l.c_str());
 char input;
 for (int i=0;i<l.size();i++)
 {
   input = line[i];
   if(isalnum(input)) //checks if the char is an operand (alphanumerical)
     cout << input;

   else //if not alphanumeric
   {
     if(input == '(')
       {s.push(input);}//'(' has lowest precedence in stack

      else if(input == ')')
      {
      while(!s.empty()&&s.top()!='(')
      {
         cout << s.top();//any operator found will be printed and popped
from stack until open parenthesis is found
         s.pop();
      }
      if(!s.empty())//any open parenthesis found will be popped off
```

```
      s.pop();
      else
      {cout << "\nproblem: '(' missing!\n";
      return 0;}//no open parenthesis indicates error and exits program
     }
     else if (s.empty()||prec(s.top()) < prec(input))
      s.push(input);//if empty or preexisting operator presiding over
input, push input to top of stack
     else if(prec(s.top()) >= prec(input))
       {
        while(!s.empty())
        {
            cout << s.top();//otherwise until stack is empty print all
data and pop them off stack
            s.pop();
        }
      s.push(input);//push in operator
      }
      }
}
while(!s.empty())//if stack is still not empty print out remaining data
 {
      cout << s.top();
      s.pop();
 }

return 0;//end program
}
```