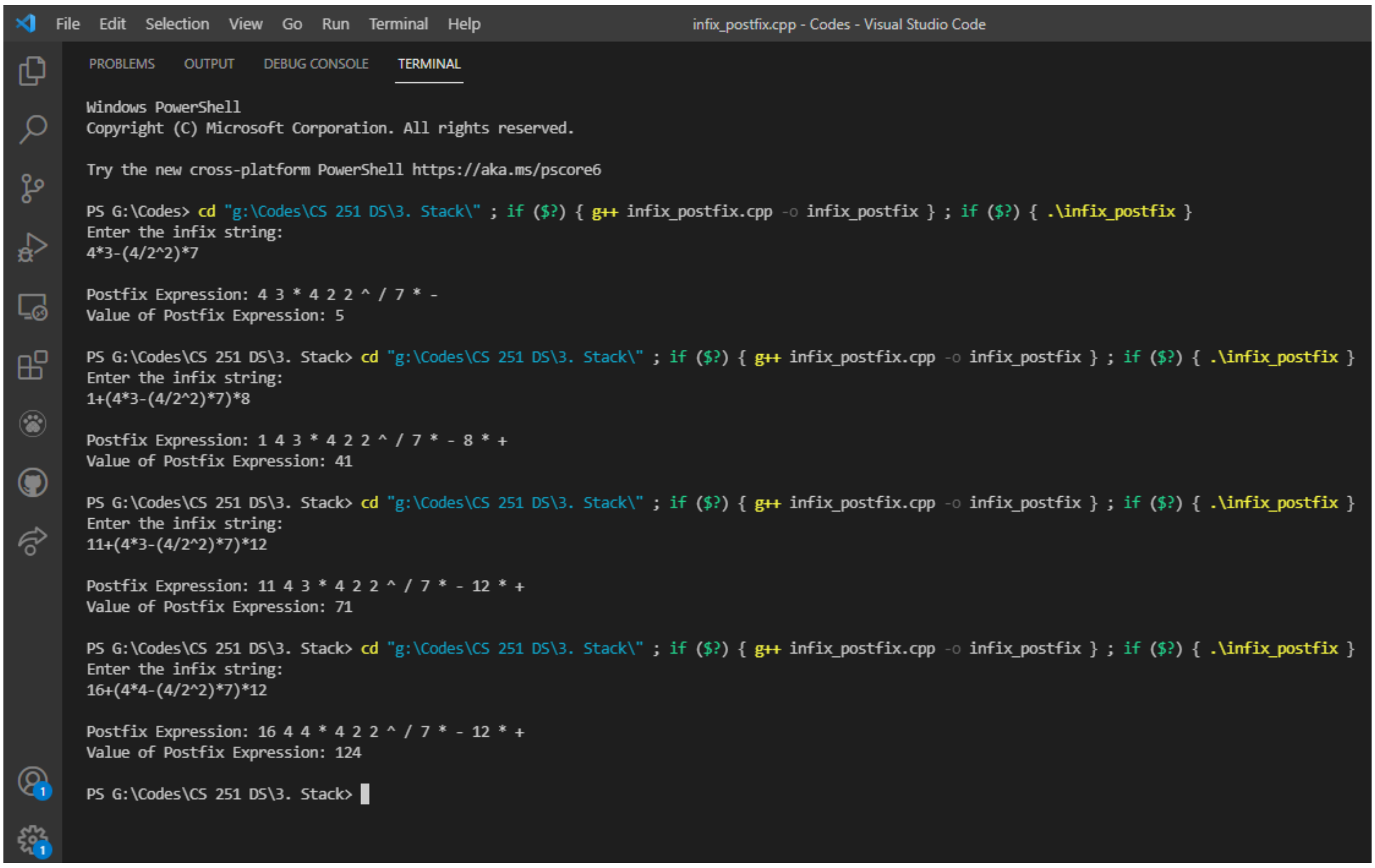
```cpp
// Aneesh Panchal
// 2K20/MC/21

#include<iostream>
#include<stack>
using namespace std;

int raise(int num,int pow)
{
    int result=1;
    for(int i=0;i<pow;++i)
        result=result*num;
    return result;
}

int precedence(char opr)
{
    if(opr=='^')
        return 3;
    else if(opr=='/' || opr=='*')
        return 2;
    else if(opr=='+' || opr=='-')
        return 1;
    else
        return -1;
}

string infix_postfix(string infix)
{
    stack<char> stak;
    string postfix;

    infix = infix + ')';
    stak.push('(');

    for(int i=0;i<infix.length();++i)
    {
        if((infix[i]>='0' && infix[i]<='9'))
            postfix+=infix[i];

        else if(infix[i]=='(')
            stak.push(infix[i]);

        else if(infix[i]==')')
        {
            while(!stak.empty() && stak.top()!='(')
            {
                postfix = postfix + ' ' + stak.top();
                stak.pop();
            }
            if(!stak.empty())
                stak.pop();
        }
        else if(infix[i]=='+' || infix[i]=='
' || infix[i]=='*' || infix[i]=='/' || infix[i]=='^')
        {
            postfix=postfix+' ';
            while(!stak.empty() && precedence(stak.top())>=precedence(infix[i]))
            {
                postfix = postfix + stak.top() + ' ';
                stak.pop();
            }
            stak.push(infix[i]);
        }

        else
            return "Error !!!";
    }
    return postfix;
}

int postfix_evaluate(string postfix)
{
    int operator1, operator2, result, input;
    stack<int> stak;
    int r;

    for(int i=0;i<postfix.length();++i)
    {
        r=i;input=0;
        if(postfix[i]>='0' && postfix[i]<='9')
        {
            while(postfix[r]!=' '){
                input=(input*10) + postfix[r]-'0';
                ++r;
            }
            i=r-1;
            stak.push(input);
        }

        else if(postfix[i]==' ')
            continue;

        else if(postfix[i]=='^')
        {
            operator2=stak.top();
            stak.pop();
            operator1=stak.top();
            stak.pop();
            result=raise(operator1,operator2);
            stak.push(result);
        }

        else if(postfix[i]=='*')
        {
            operator2=stak.top();
            stak.pop();
            operator1=stak.top();
            stak.pop();
            result=operator1*operator2;
            stak.push(result);
        }

        else if(postfix[i]=='/')
        {
            operator2=stak.top();
            stak.pop();
            operator1=stak.top();
            stak.pop();
            result=operator1/operator2;
            stak.push(result);
        }

        else if(postfix[i]=='+')
        {
            operator2=stak.top();
            stak.pop();
            operator1=stak.top();
            stak.pop();
            result=operator1+operator2;
            stak.push(result);
        }

        else if(postfix[i]=='-')
        {
            operator2=stak.top();
            stak.pop();
            operator1=stak.top();
            stak.pop();
            result=operator1-operator2;
            stak.push(result);
        }

        else
            return -1;
    }
    return stak.top();
}

int main()
{
    // 4*3-(4/2^2)*7 = 5
    // 1+(4*3-(4/2^2)*7)*8 = 41
    // 11+(4*3-(4/2^2)*7)*12 = 71

    string infix;
    cout<<"Enter the infix string:"<<endl;
    getline(cin,infix);

    string postfix = infix_postfix(infix);
```

```cpp
    int result = postfix_evaluate(postfix);

    cout<<endl<<"Postfix Expression: "<<postfix<<endl;

    if(result==-1)
        cout<<"Value of Postfix Expression: Error !!!"<<endl<<endl;
    else
        cout<<"Value of Postfix Expression: "<<result<<endl<<endl;

    return 0;
}
```


File Edit Selection View Go Run Terminal Help
infix_postfix.cpp - Codes - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS G:\Codes> cd "g:\Codes\CS 251 DS\3. Stack\" ; if ($?) { g++ infix_postfix.cpp -o infix_postfix } ; if ($?) { .\infix_postfix }
Enter the infix string:
4*3-(4/2^2)*7

Postfix Expression: 4 3 * 4 2 2 ^ / 7 * -
Value of Postfix Expression: 5

PS G:\Codes\CS 251 DS\3. Stack> cd "g:\Codes\CS 251 DS\3. Stack\" ; if ($?) { g++ infix_postfix.cpp -o infix_postfix } ; if ($?) { .\infix_postfix }
Enter the infix string:
1+(4*3-(4/2^2)*7)*8

Postfix Expression: 1 4 3 * 4 2 2 ^ / 7 * - 8 * +
Value of Postfix Expression: 41

PS G:\Codes\CS 251 DS\3. Stack> cd "g:\Codes\CS 251 DS\3. Stack\" ; if ($?) { g++ infix_postfix.cpp -o infix_postfix } ; if ($?) { .\infix_postfix }
Enter the infix string:
11+(4*3-(4/2^2)*7)*12

Postfix Expression: 11 4 3 * 4 2 2 ^ / 7 * - 12 * +
Value of Postfix Expression: 71

PS G:\Codes\CS 251 DS\3. Stack> cd "g:\Codes\CS 251 DS\3. Stack\" ; if ($?) { g++ infix_postfix.cpp -o infix_postfix } ; if ($?) { .\infix_postfix }
Enter the infix string:
16+(4*4-(4/2^2)*7)*12

Postfix Expression: 16 4 4 * 4 2 2 ^ / 7 * - 12 * +
Value of Postfix Expression: 124

PS G:\Codes\CS 251 DS\3. Stack> 
```