```cpp
#include <iostream>
#include "operand.h"
#include "stack.h"
#include <math.h>

using namespace std;

#ifndef expr_float_h
#define expr_float_h

int expr_float(){

    cout << "Enter a Numeric Expression ( May include integers,(),*,/,%,^,-,+ ).";
    while(true){

        int MAXLEN(200);
        char* raw(new char[MAXLEN]);           // creating a char Array to
        cout << "\n[float]> ";                   // store user input

        cin.getline( raw , MAXLEN-1 , '\n' );  // taking input from user
        if(!strnlen(raw,MAXLEN)){              // Quit if no input
            return 0;
        }

        stack<double> postfix;                 // creating a double stack
        stack<operand> opd;                    // creating an operand stack

        opd.push_back(operand(-1,'('));        // Pushing an opening bracket

        bool error(0);                         // an error flag

        /* Following loop converts Expression to
         * postfix and calculates it: */
        for( int i=0, iflag(0), dflag(0); i<=strlen(raw) ; ++i ){

            //1. For a Literal
            if((int)(raw[i])-48 >= 0 && (int)(raw[i])-48 <= 9 || raw[i]=='.'){
                if(iflag && !dflag){
                    if(!(raw[i]=='.')){
                        double a =(float)((int)(raw[i])-48) + postfix.top() * 10;
                        postfix.pop_out();
                        dflag=0;
                        postfix.push_back(a);
                    }
                    else if(raw[i]=='.'){
                        dflag=1;
                    }

                }
                else if(iflag && dflag){
                    if(!(raw[i]=='.')){
                        double a =(float)((int)(raw[i])-48)/(pow(10,dflag++))
                            + postfix.top();
                        postfix.pop_out();
                        postfix.push_back(a);
                    }
                    else if(raw[i]=='.'){
                        error=1;
                        cout << "-> Invalid String" << endl;
                        break;
                    }
                }
                else if(!iflag && !dflag){
                    iflag=1;
                    int a =(int)(raw[i])-48;
                    postfix.push_back(a);
                }
                else if(!iflag && dflag){
```

```
                error=1;
                cout << "-> Invalid String" << endl;
                break;
            }
        }

        //2. For an Operand
        else if(raw[i] == '(' || raw[i] == ')' ||
                raw[i] == '*' || raw[i] == '/' ||
                raw[i] == '%' || raw[i] == '-' ||
                raw[i] == '+' || raw[i] == '^' ||
                raw[i] == ' ' || raw[i] == '\0'){
            iflag=0;dflag=0;
            int poco;
            switch(raw[i]){
                case '+':case '-':poco=1;break;
                case '*':case '/':case '%':poco=2;break;
                case '^':poco=3;break;
                case ')':poco=-2;break;
                case '(':poco=-1;break;
                default: poco=0;break;
            }
            operand dob(poco,raw[i]);
            if( (dob > 0 && dob >= opd.top()) || dob == -1 ){
                opd.push_back(dob);
            }
            else if( dob > 0 && dob <  opd.top()){
                operand poped(opd.top().p, opd.top().o);
                while(dob < poped){
                    opd.pop_out();
                    double b = postfix.get(postfix.size());
                    double a = postfix.get(postfix.size()-1);
                    double r(1);
                    postfix.pop_out();
                    postfix.pop_out();
                    switch(poped.o){
                        case '+':r=a+b;break;
                        case '-':r=a-b;break;
                        case '*':r=a*b;break;
                        case '/':r=a/b;break;
                        case '%':r=fmod(a,b);break;
                        case '^':r=pow(a,b);break;
                        default: r=a+b;break;
                    }
                    postfix.push_back(r);
                    poped(opd.top().p, opd.top().o);
                }
                opd.push_back(dob);
            }
            else if(dob == -2 || dob.o == '\0'){
                operand poped(opd.top().p, opd.top().o);
                while(poped != -1){
                    opd.pop_out();
                    double b = postfix.get(postfix.size());
                    double a = postfix.get(postfix.size()-1);
                    double r(1);
                    postfix.pop_out();
                    postfix.pop_out();
                    switch(poped.o){
                        case '+':r=a+b;break;
                        case '-':r=a-b;break;
                        case '*':r=a*b;break;
                        case '/':r=a/b;break;
                        case '%':r=fmod(a,b);break;
                        case '^':r=pow(a,b);break;
                        default: r=a+b;break;
                    }
                    postfix.push_back(r);
```

```
                        poped(opd.top().p, opd.top().o);
                    }
                    opd.pop_out();
                }

            }
            //else
            else{
                error=1;
                cout << "-> Invalid String" << endl;
                break;
            }
        }
    if(!error)
    cout << "=> Answer: " << postfix.top();
    postfix.clear();
    opd.clear();
    cout << endl ;

    }
    return 0;
}

#endif
```