11) EVALUATION OF ARITHMETIC EXPRESSIONS

This is the standard problems of stack:

INFIX TO POSTFIX:

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

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

bool operate(char c){
    if(c=='+'||c=='-'||c=='*'||c=='/'||c=='('||c==')'){
        return true;
    }
    else{
        return false;
    }
}

string infixToPostfix(string st){
    stack<char> s;
    string ans="";
    for(int i=0;i<st.length();i++){
        if(operate(st[i])==false){
            ans=ans+st[i];
        }
        else{
```

```
            if(s.size()==0){
                s.push(st[i]);
            }
            else{
                if(st[i]==')'){
                    while(s.size()!=0 && s.top()!='('){
                        ans=ans+s.top();
                        s.pop();
                    }
                    s.pop();
                }
                else if(s.size()!=0 && s.top()=='('){
                    s.push(st[i]);
                }
                else if(prior(s.top())<prior(st[i])){
                    s.push(st[i]);
                }
                else{
                    while(s.size()!=0 && prior(s.top())>=prior(st[i])){
                        ans=ans+s.top();
                        s.pop();
                        if(s.size()!=0 && s.top()=='('){
                            break;
                        }
                    }
                    s.push(st[i]);
                }
            }
        }
    }
    while(s.size()!=0){
        ans=ans+s.top();
        s.pop();
    }
    return ans;
}

int main(){
    string st;
```

```
    cout<<"\n Enter the expression to be evaluated in the infix form:";
    cin>>st;
    string answer=infixToPostfix(st);
    cout<<"\n The PostFix of the above Expression:"<<answer;
    return 0;
}
```

INFIX TO PREFIX:

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

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

bool operate(char c){
    if(c=='+'||c=='-'||c=='*'||c=='/'||c=='('||c==')'){
        return true;
    }
    else{
        return false;
    }
}

string infixToPrefix(string st){
    stack<char> s;
    string ans="";
```

```
    reverse(st.begin(),st.end());
    for(int i=0;i<st.length();i++){
        if(operate(st[i])==false){
            ans=ans+st[i];
        }
        else{
            if(s.size()==0){
                s.push(st[i]);
            }
            else{
                if(st[i]=='('){
                    while(s.size()!=0 && s.top()!=')'){
                        ans=ans+s.top();
                        s.pop();
                    }
                    s.pop();
                }
                else if(s.top()==')'){
                    s.push(st[i]);
                }
                else if(prior(s.top())<=prior(st[i])){
                    s.push(st[i]);
                }
                else{
                    while(s.size()!=0 && prior(s.top())>prior(st[i])){
                        ans=ans+s.top();
                        s.pop();
                        if(s.size()!=0 && s.top()==')'){
                            break;
                        }
                    }
                    s.push(st[i]);
                }
            }
        }
    }
    while(s.size()!=0){
        ans=ans+s.top();
        s.pop();
```

```
    }
    reverse(ans.begin(),ans.end());
    return ans;
}

int main(){
    string st;
    cout<<"\n Enter the expression to be evaluated in the infix form:";
    cin>>st;
    string answer=infixToPrefix(st);
    cout<<"\n The PreFix of the above Expression:"<<answer;
    return 0;
}
```

POSTFIX TO INFIX:

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

bool operate(char c){
    if(c=='+'||c=='-'||c=='*'||c=='/'||c=='('||c==')'){
        return true;
    }
    else{
        return false;
    }
}

string postfixToInfix(string st){
    stack<string> s;
    for(int i=0;i<st.length();i++){
        if(operate(st[i])==true){
            string t=st[i]+s.top()+')';
            s.pop();
            t='('+s.top()+t;
```

```
            s.pop();
            s.push(t);
        }
        else{
            s.push(to_string(st[i]));
        }
    }
    return s.top();
}

int main(){
    string st;
    cout<<"\n Enter the expression to be evaluated in the postfix
form:";
    cin>>st;
    string answer=postfixToInfix(st);
    cout<<"\n The PreFix of the above Expression:"<<answer;
    return 0;
}
```

POSTFIX TO INFIX:

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

bool operate(char c){
    if(c=='+'||c=='-'||c=='*'||c=='/'||c=='('||c==')'){
        return true;
    }
    else{
        return false;
    }
}

string postfixToInfix(string st){
```

```
    reverse(st.begin(),st.end());
    stack<string> s;
    for(int i=0;i<st.length();i++){
        if(operate(st[i])==true){
            string t='('+s.top()+st[i];
            s.pop();
            t=t+s.top()+')';
            s.pop();
            s.push(t);
        }
        else{
            s.push(to_string(st[i]));
        }
    }
    return s.top();
}

int main(){
    string st;
    cout<<"\n Enter the expression to be evaluated in the postfix
form:";
    cin>>st;
    string answer=postfixToInfix(st);
    cout<<"\n The PreFix of the above Expression:"<<answer;
    return 0;
}
```