DSA_College\infixToPrefix.cpp

```cpp
// [infix to prefix]
#include<bits/stdc++.h>
using namespace std ;

int precedence(char x)
{
    if(x == '+' || x == '-')
    return 1 ;

    if(x == '*' || x == '/')
    return 2 ;

    if(x == '^')
    return 3 ;

    else return -1 ; // for non-operators
}

bool isOperand(char x)
{
    return ((x >= 'a' && x <= 'z') || (x >= 'A' && x <= 'Z')) ;
}

string infixToPrefix(string infix)
{
    stack<char> st ;
    string prefix = "" ;

    // step1 - reverse the infix expression
    reverse(infix.begin(),infix.end()) ;

    // step2 - scan from left to right
     for(int i=0;i<infix.length();i++)
     {
        // if operand comes, push to output string
        char ch = infix[i] ;
        if(isOperand(ch))
        {
            prefix += ch ;
        }

        // ignore whitespaces
        else if(isspace(ch))
        continue ;

        // if closing bracket ) comes, push to stack
        else if(ch == ')')
        {
            st.push(')') ;
        }

```

```
        // if opening bracket comes, pop from stack until opening bracket comes
        // and push the popped oerators into final string
        else if(ch == '(')
        {
            while(st.top() != ')')
            {
                prefix += st.top();
                st.pop() ;
            }
            st.pop() ;
        }

        // operator comes
        // if new operator comes has precedence greater than or equal to stack top operator,
        // then push the new operator to stack, else pop it
        else{
            while(!st.empty() && precedence(st.top()) > precedence(ch))
            {
              prefix += st.top();
              st.pop();
            }
            st.push(ch) ;
            }
     }

     //pop all operators from stack
     while(!st.empty())
     {
        prefix += st.top();
        st.pop() ;
     }

     // reverse the output string come
     reverse(prefix.begin(),prefix.end()) ;

     return prefix ;
}
int main()
{
    string infix = "((a+(b*c))-d)" ;
    cout<<infixToPrefix(infix)<<endl;
    return 0;
}
```