Pastebin Link: <https://paste.ubuntu.com/p/NgJjtn2n7H/>

#define NUMBER\_OF\_OPERATORS 3  
**using** **namespace** std;  
  
  
**char** op[]={'\*', '.', '|'}; */\*List all operators in*  
 *decreasing order of*  
 *precedence\*/*  
  
**inline** **int** operator\_precedence(**char** ch){  
 **for**(**int** i=0; i<NUMBER\_OF\_OPERATORS; i++) **if**(ch==op[i]) **return** i;  
}  
  
**inline** **bool** isOperator(**char** ch){  
 **for**(**int** i=0; i<NUMBER\_OF\_OPERATORS; i++){  
 **if**(ch==op[i]) **return** **true**;  
 }  
 **return** **false**;  
}  
  
stack <**char**> opstack;  
  
**void** infixToPostfix(**char** postFix[], **int** &postlen, **char** infix[], **int** &inlen){  
 **while**(!opstack.empty()) opstack.pop();  
 postlen=0;  
 **for**(**int** i=0; i<inlen; i++){  
 **if**(infix[i]=='('){  
 opstack.push('(');  
 }  
 **else** **if**(infix[i]==')'){  
 **while**(opstack.top()!='('){  
 postFix[postlen]=opstack.top();  
 postlen++;  
 opstack.pop();  
 }  
 opstack.pop();  
 }  
 **else** **if**(!isOperator(infix[i])){  
 postFix[postlen]=infix[i];  
 postlen++;  
 }  
 **else**{  
 **while**(!opstack.empty() && operator\_precedence(opstack.top())<=operator\_precedence(infix[i])){  
 postFix[postlen]=opstack.top();  
 postlen++;  
 opstack.pop();  
 }  
 opstack.push(infix[i]);  
 }  
 }  
 **while**(!opstack.empty()){  
 postFix[postlen]=opstack.top();  
 postlen++;  
 opstack.pop();  
 }  
 postFix[postlen]=0;  
}