```
#include<iostream>
 #include<cctype>
using namespace std;
 //Node for the linked kist stack
 struct Node{
    char data;
    Node*next;
};
 //Stack class using singly linked list
 class Stack{
    Node*top;
public:
    Stack(){
     top=NULL;
 }
 //Push a character onto the stack
 void push(char x){
    Node*t = new Node;
    t->data = x;
    t->next = top;
    top=t;
 }
 //Pop a character from the stack
 char pop(){
   if(top == NULL) return'\0';
   char x = top->data;
   Node*t=top;
   top = top->next;
```

```
  delete t;
  return x;
}
//Peek at the top of stack
char peek(){
  if(top != NULL)return top->data;
  else return'\0';
}
//check if stack is empty
bool isEmpty(){
  return top==NULL;
 }
};
//Function to check precedence of operators
int precedence(char op){
  if(op=='+'||op=='-')return 1;
  if(op=='*'||op=='/')return 2;
  return 0;
}
//Function to convert infix expression to postfix
string infixToPostfix(string infix){
   Stack s;
   string postfix="";
   for(char ch :infix){
     if(isalnum(ch)){
      postfix +=ch;
      }
      else if(ch=='('){
```

```
      s.push(ch);
      }
      else if (ch==')'){
        while(!s.isEmpty()&& s.peek()!='(')
          postfix +=s.pop();
        s.pop();
      }
      else{
          while(!s.isEmpty()&& precedence(ch)<=precedence(s.peek()))
          postfix +=s.pop();
         s.push(ch);
      }
     }
     //pop remaning operators from stack
     while(!s.isEmpty())
         postfix +=s.pop();
     return postfix;
   }
string infixToPrefix(string infix){
   string rev="";
   for(int i=infix.length()-1;i>=0;i--){
      if(infix[i]=='(')
         rev += ')';
      else if(infix[i]==')')
         rev += '(';
      else
         rev += infix[i];
    }
    string postfix = infixToPostfix(rev);
    string prefix ="";
    for(int i=postfix.length()-1;i>=0;i--){
      prefix += postfix[i];
    }
    return prefix;
}
int main(){
  string infix;
  //Get infix input from user
  cout<<"Enter infix:";
  cin>>infix;
  //Call functions and show output
  cout<<"postfix:"<<infixToPostfix(infix)<<endl;
  cout<<"prefix:"<<infixToPrefix(infix)<<endl;
  return 0;
}
```