#include <iostream>

#include <algorithm>

using namespace std;

const int STACK\_SIZE=100;

template <typename T>

class Stack {

private:

int top;

T arr[STACK\_SIZE];

public:

Stack(){

top=-1;

}

void push(T elem){

if(top<STACK\_SIZE){

arr[++top]=elem;

}

else{

cout<<"Overflow"<<endl;

}

}

T pop(){

if(top>=0)

return arr[top--];

}

T peek(){

if(top>=0)

return arr[top];

}

bool isEmpty(){

if(top==-1){

return true;

}

return false;

}

};

string pretoInfix(string pre){

Stack<string> s;

string output, final;

reverse(pre.begin(),pre.end());

int l=pre.length();

for(int i=0;i<l;i++){

if(isdigit(pre[i])||isalpha(pre[i])){

s.push(string(1,pre[i]));

}

else{

string a=s.pop();

string b=s.pop();

string exp='('+b+pre[i]+a+')';

s.push(exp);

}

}

output=s.peek();

int len=output.length();

for(int i=len-1;i>=0;i--){

if(output[i]=='('){

final+=')';}

else if(output[i]==')'){

final+='(';}

else

{

final+=output[i];

}

}

return final;

}

int getPriority(char c){

switch (c)

{

case '-':

case '+':

return 1;

break;

case '\*':

case '/':

return 2;

break;

case '$':

case '^':

return 3;

break;

default:

return 0;

break;

}

}

string infixtoPostfix(string infix){

infix='('+infix+')';

int l=infix.size();

string output;

Stack<char> s;

for(int i=0;i<l;i++){

if(isalpha(infix[i])||isdigit(infix[i]))

output+=infix[i];

else if(infix[i]=='(')

s.push(infix[i]);

else if(infix[i]==')'){

while(s.peek()!='(')

output+=s.pop();

s.pop();

}

else{

while(getPriority(infix[i])<=getPriority(s.peek()))

output+=s.pop();

s.push(infix[i]);

}

}

return output;

}

int main(){

string inp;

cout<<"Enter the expression:";

cin>>inp;

cout<<infixtoPostfix(pretoInfix(inp));

}