**Algorithm**

1. Push ‘(‘ to the stack.

2. Scan the infix expression from left to right and repeat the steps 3 to 6 for each element until the stack is empty.

3. If an operand is encountered, add it to the postfix expression.

4. If a left parenthesis is encountered, then push it into the stack.

5. If an operator is encountered, then:

(a) Repeatedly pop all the operators from the stack which have the same or higher precedency than the operator that is being encountered and add it to the postfix expression.

(b) Push the operator to the satck.

6. If right parenthesis is encountered, then:

(a) Repeatedly pop operators from stack and add it to the postfix expression until a left parenthesis is encountered.

(b) Remove the left parenthesis and pop it.

7. Stop.

**Program Code**

#include<iostream>

using namespace std;

class st

{

int top;

char a[20];

int size;

public:

st()

{

top=-1;

size=20;

}

void push(char x)

{

if(top>=size-1)

{

cout<<"Stack overflow";

}

else

{

top=top+1;

a[top]=x;

}

}

char pop()

{

if(top<0)

{

cout<<"Stack underflow";

}

else

{

char c;

c=a[top];

top--;

return c;

}

}

int pre(char s)

{

if(s == '^')

{

return(3);

}

else if(s== '\*' || s == '/')

{

return(2);

}

else if(s == '+' || s == '-')

{

return(1);

}

else

{

return(0);

}

}

void conv(char exp[])

{

int i=0,j=0;

char item,x;

char ans[20];

item=exp[0];

while(item!='\0')

{

if(item=='(')

{

push(item);

}

else if(isalpha(item)||isdigit(item))

{

ans[j]=item;

j++;

}

else if(item == '^' || item== '\*' || item == '/' || item == '+' || item=='-')

{

x=pop();

while((item == '^' || item== '\*' || item == '/' || item == '+' || item=='-')&&pre(x)>=pre(item))

{

ans[j]=x;

j++;

x=pop();

}

push(x);

push(item);

}

else if(item==')')

{

x=pop();

while(x!='(')

{

ans[j]=x;

j++;

x=pop();

}

}

i++;

item=exp[i];

}

cout<<”\nPostfix exp=”<<ans;

}

};

int main()

{

st obj;

char exp[20];

cout<<"Enter the exp";

cin>>exp;

obj.conv(exp);

return 0;

}

**Output**

Enter the exp (a+b\*c)

Postfix exp= abc\*+

Enter the exp (a\*b+c)

Postfix exp= ab\*c+