//Write a program to convert an infix expression to its corresponding prefix expressions.

# include <stdio.h>

# include <string.h>

# define MAX 20

void infixtoprefix(char infix[20],char prefix[20]);

void reverse(char array[30]);

char pop();

void push(char symbol);

int isOperator(char symbol);

int prcd(symbol);

int top=-1;

char stack[MAX];

main() {

char infix[20],prefix[20],temp;

printf("Enter infix operation: ");

gets(infix);

infixtoprefix(infix,prefix);

reverse(prefix);

puts((prefix));

}

//--------------------------------------------------------

void infixtoprefix(char infix[20],char prefix[20]) {

int i,j=0;

char symbol;

stack[++top]='#';

reverse(infix);

for (i=0;i<strlen(infix);i++) {

symbol=infix[i];

if (isOperator(symbol)==0) {

prefix[j]=symbol;

j++;

} else {

if (symbol==')') {

push(symbol);

} else if(symbol == '(') {

while (stack[top]!=')') {

prefix[j]=pop();

j++;

}

pop();

} else {

if (prcd(stack[top])<=prcd(symbol)) {

push(symbol);

} else {

while(prcd(stack[top])>=prcd(symbol)) {

prefix[j]=pop();

j++;

}

push(symbol);

}

//end for else

}

}

//end for else

}

//end for for

while (stack[top]!='#') {

prefix[j]=pop();

j++;

}

prefix[j]='\0';

}

////--------------------------------------------------------

void reverse(char array[30])

{// for reverse of the given expression {

int i,j;

char temp[100];

for (i=strlen(array)-1,j=0;i+1!=0;--i,++j) {

temp[j]=array[i];

}

temp[j]='\0';

strcpy(array,temp);

return array;

}

//--------------------------------

char pop() {

char a;

a=stack[top];

top--;

return a;

}

//----------------------------------

void push(char symbol) {

top++;

stack[top]=symbol;

}

//------------------------------------------

int prcd(symbol)

{// returns the value that helps in the precedence {

switch(symbol) {

case '+':

case '-':

return 2;

break;

case '\*':

case '/':

return 4;

break;

case '$':

case '^':

return 6;

break;

case '#':

case '(':

case ')':

return 1;

break;

}

}

//-------------------------------------------------

int isOperator(char symbol) {

switch(symbol) {

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '&':

case '(':

case ')':

return 1;

break;

default:

return 0;

// returns 0 if the symbol is other than given above

}

}