***// INFIX TO POSTFIX CONVERSION***

#include<stdio.h>

#include<ctype.h>

#define MAX 50

typedef struct stack

{

int data[MAX];

int top;

}stack;

int precedence(char);

void init(stack \*);

int empty(stack \*);

int full(stack \*);

int pop(stack \*);

void push(stack \*,int);

int top(stack \*); *//value of the top element*

void infix\_to\_postfix(char infix[],char postfix[]);

int main()

{

char infix[30],postfix[30];

printf("Enter an infix expression (do not leave space between characters): ");

gets(infix);

infix\_to\_postfix(infix,postfix);

printf("\nPostfix expression: %s\n",postfix);

return 0;

}

void infix\_to\_postfix(char infix[],char postfix[])

{

stack s;

char x,token;

int i,j; *//i-index of infix,j-index of postfix*

init(&s);

j=0;

for(i=0;infix[i]!='\0';i++)

{

token=infix[i];

if(isalnum(token))

postfix[j++]=token;

else if(token=='(')

push(&s,'(');

else if(token==')')

while((x=pop(&s))!='(')

postfix[j++]=x;

else

{

while(precedence(token)<=precedence(top(&s))&&!empty(&s))

{

x=pop(&s);

postfix[j++]=x;

}

push(&s,token);

}

}

while(!empty(&s))

{

x=pop(&s);

postfix[j++]=x;

}

postfix[j]='\0';

}

int precedence(char x)

{

if(x=='(')

return(0);

if(x=='+'||x=='-')

return(1);

if(x=='\*'||x=='/'||x=='%')

return(2);

return(3);

}

void init(stack \*s)

{

s->top=-1;

}

int empty(stack \*s)

{

if(s->top==-1)

return(1);

return(0);

}

int full(stack \*s)

{

if(s->top==MAX-1)

return(1);

return(0);

}

void push(stack \*s,int x)

{

s->top=s->top+1;

s->data[s->top]=x;

}

int pop(stack \*s)

{

int x;

x=s->data[s->top];

s->top=s->top-1;

return(x);

}

int top(stack \*p)

{

return (p->data[p->top]);

}