#include<stdio.h>

#include<string.h>

#include<math.h>

#include<malloc.h>

#include<stdlib.h>

#define BLANK ' '

#define MAX 50

#define TAB '\t'

//function declarations

void push(long int symbol);

long int pop();

void infix\_to\_postfix();

int priority(char symbol);

int isEmpty();

int white\_space(char);

char infix[MAX],postfix[MAX];

long int stack[MAX];

int top;

int main()

{

long int value;

top=-1;

printf("Enter infix : ");

gets(infix);

infix\_to\_postfix();

printf("Postfix : ");

for(int i=0;i<strlen(postfix);i++)

printf("%c ",postfix[i]);

return 0;

}

void infix\_to\_postfix()//converts infix to postfix

{

unsigned int i,p=0;

char next;

char symbol;

//iterates through infix

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

{

symbol=infix[i];

if(!white\_space(symbol)) //no whitespace

{

switch(symbol)

{

case '(': push(symbol);

break;

case ')':

while((next=pop())!='(')

postfix[p++]=next;

break;

case '+':

case '-':

case '\*':

case '/':

case '%':

case '^':

while(!isEmpty()&& priority(stack[top])>=priority(symbol))

postfix[p++]=pop();

push(symbol);

break;

default:postfix[p++]=symbol;

}

}

}

while(!isEmpty())

postfix[p++]=pop();

postfix[p]='\0';

}

int priority(char symbol)//checks the relative priority of symbols

{

switch(symbol)

{

case '(':return 0;

case '+':

case '-':return 1;

case '\*':

case '/':

case '%':

return 2;

case '^':

return 3;

default:

return 0;

}

}

void push(long int symbol) //helper push function to push elements in postfix

{

if(top>MAX)

{

printf("Stack overflow\n");

exit(1);

}

stack[++top]=symbol;

}

long int pop() //helper pop function to pop elements in postfix

{

if(isEmpty())

{

printf("Stack underflow\n");

exit(1);

}

return (stack[top--]);

}

int isEmpty() //checks if stack in empty

{

if(top==-1)

{

return 1;

}

else

{

return 0;

}

}

int white\_space(char symbol) //checks for whitespaces or tabs

{

if(symbol==BLANK||symbol==TAB)

return 1;

else

return 0;

}

OUTPUT:

Enter infix : 5+6-8\*(7-3)+4/2

Postfix : 5 6 + 8 7 3 - \* - 4 2 / +