#include<stdio.h>

#include<ctype.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

#define MAX 100

#define SIZE 100

int top = -1;

int stackC[MAX];

char stackD[20];

char stackE[20],infx[20],prfx[20];

void pushC(int x);

int popC();

void EvalPostfix(char postfix[]);

char popD();

void pushD(char x);

int priority(char x);

char popE();

void pushE(char symbol);

void reverse(char a[30]);

void infxtoprfx(char infx[20],char prfx[20]);

int op(char symbol);

int prcd(symbol);

int main()

{

int choice;

int i;

char postfix[SIZE];

char expr[20];

char \*e, x;

while(1)

{

printf("\n1.Postfix evaluation\n");

printf("2.Infix to postfix evaluation\n");

printf("3.Infix to prefix evaluation\n");

printf("\nEnter a choice:");

scanf("%d",&choice);

switch(choice)

{

case 1:printf( " \npress right parenthesis ')' for end of expression,\nEnter postfix expression:\n ");

for (i = 0 ; i <= SIZE - 1 ; i++)

{

scanf("%c", &postfix[i]);

if ( postfix[i] == ')' )

{ break; }

}

EvalPostfix(postfix);

break;

case 2:printf("Enter the expression: ");

scanf("%s",expr);

e = expr;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c",\*e);

else if(\*e == '(')

pushD(\*e);

else if(\*e == ')')

{

while((x = popD()) != '(')

printf("%c", x);

}

else

{

while(priority(stackD[top]) >= priority(\*e))

printf("%c",popD());

pushD(\*e);

}

e++;

}

while(top != -1)

{

printf("%c",popD());

}

break;

case 3:printf("Enter infix expression:");

gets(infx);

infxtoprfx(infx,prfx);

reverse(prfx);

puts((prfx));

break;

default:printf("\nInvalid choice\n");

break;

}

}

}

void pushC(int x)

{

if(top >= MAX -1)

{

printf("stack over flow");

return;

}

else

{

top = top + 1 ;

stackC[top]= x;

}

}

int popC()

{

int x;

if(top <0)

{

printf("stack under flow");

}

else

{

x = stackC[top];

top = top - 1;

return x;

}

}

void EvalPostfix(char postfix[])

{

int i;

char ch;

int c;

int a,b;

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

{

ch = postfix[i];

if (isdigit(ch))

{

pushC(ch-'0');

}

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

{

a = popC();

b = popC();

switch (ch)

{

case '\*':c = b \* a;

break;

case '/':c = b / a;

break;

case '+':c = b + a;

break;

case '-':c = b - a;

break;

}

pushC(c);

}

}

printf( "\nResult of expression evaluation : %d \n", popC()) ;

}

////////////////////////////////////////////////////////////////////////////////////////////////

void pushD(char x)

{

stackD[++top] = x;

}

char popD()

{

if(top == -1)

return -1;

else

return stackD[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

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

return 1;

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

return 2;

}

////////////////////////////////////////////////////////////////////////////////////////////////

void infxtoprfx(char infx[20],char prfx[20]) {

int i,j=0;

char symbol;

stackE[++top]='#';

reverse(infx);

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

{

symbol=infx[i];

if (op(symbol)==0)

{

prfx[j]=symbol;

j++;

} else

{

if (symbol==')')

{

pushE(symbol);

} else if(symbol == '(')

{

while (stackE[top]!=')')

{

prfx[j]=popE();

j++;

}

popE();

} else

{

if (prcd(stackE[top])<=prcd(symbol))

{

pushE(symbol);

} else

{

while(prcd(stackE[top])>=prcd(symbol))

{

prfx[j]=popE();

j++;

}

pushE(symbol);

}

}

}

}

while (stackE[top]!='#')

{

prfx[j]=pop();

j++;

}

prfx[j]='\0';

}

void reverse(char array[30])

{

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=stackE[top];

top--;

return a;

}

void pushE(char symbol)

{

top++;

stackE[top]=symbol;

}

int prcd(symbol)

{

switch(symbol)

{

case '+':

case '-':

return 2;

break;

case '\*':

case '/':

return 4;

break;

case '$':

case '^':

return 6;

break;

case '#':

case '(':

case ')':

return 1;

break;

}

}

int op(char symbol)

{

switch(symbol)

{

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '&':

case '(':

case ')':

return 1;

break;

default:

return 0;

}

}

///////////////////

#include<stdio.h>

#include<string.h>

#define MAX 20

char popE();

void pushE(char symbol);

void reverse(char a[30]);

void infxtoprfx(char infx[20],char prfx[20]);

int op(char symbol);

int prcd(symbol);

int top=-1;

char stack[MAX];

int main()

{

char infx[20],prfx[20],temp;

printf("Enter infix expression:");

gets(infx);

infxtoprfx(infx,prfx);

reverse(prfx);

puts((prfx));

}

void infxtoprfx(char infx[20],char prfx[20]) {

int i,j=0;

char symbol;

stackE[++top]='#';

reverse(infx);

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

{

symbol=infx[i];

if (op(symbol)==0)

{

prfx[j]=symbol;

j++;

} else

{

if (symbol==')')

{

pushE(symbol);

} else if(symbol == '(')

{

while (stackE[top]!=')')

{

prfx[j]=popE();

j++;

}

popE();

} else

{

if (prcd(stackE[top])<=prcd(symbol))

{

pushE(symbol);

} else

{

while(prcd(stackE[top])>=prcd(symbol))

{

prfx[j]=popE();

j++;

}

pushE(symbol);

}

}

}

}

while (stackE[top]!='#')

{

prfx[j]=pop();

j++;

}

prfx[j]='\0';

}

void reverse(char array[30])

{

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=stackE[top];

top--;

return a;

}

void pushE(char symbol)

{

top++;

stackE[top]=symbol;

}

int prcd(symbol)

{

switch(symbol)

{

case '+':

case '-':

return 2;

break;

case '\*':

case '/':

return 4;

break;

case '$':

case '^':

return 6;

break;

case '#':

case '(':

case ')':

return 1;

break;

}

}

int op(char symbol)

{

switch(symbol)

{

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '&':

case '(':

case ')':

return 1;

break;

default:

return 0;

}

}

#include<stdio.h>

#include<string.h>

#define MAX 20

char popE();

void pushE(char symbol);

void reverse(char a[30]);

void infxtoprfx(char infx[20],char prfx[20]);

int op(char symbol);

int prcd(symbol);

int top=-1;

char stack[MAX];

int main()

{

char infx[20],prfx[20],temp;

printf("Enter infix expression:");

gets(infx);

infxtoprfx(infx,prfx);

reverse(prfx);

puts((prfx));

}

void infxtoprfx(char infx[20],char prfx[20]) {

int i,j=0;

char symbol;

stackE[++top]='#';

reverse(infx);

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

{

symbol=infx[i];

if (op(symbol)==0)

{

prfx[j]=symbol;

j++;

} else

{

if (symbol==')')

{

pushE(symbol);

} else if(symbol == '(')

{

while (stackE[top]!=')')

{

prfx[j]=popE();

j++;

}

popE();

} else

{

if (prcd(stackE[top])<=prcd(symbol))

{

pushE(symbol);

} else

{

while(prcd(stackE[top])>=prcd(symbol))

{

prfx[j]=popE();

j++;

}

pushE(symbol);

}

}

}

}

while (stackE[top]!='#')

{

prfx[j]=pop();

j++;

}

prfx[j]='\0';

}

void reverse(char array[30])

{

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=stackE[top];

top--;

return a;

}

void pushE(char symbol)

{

top++;

stackE[top]=symbol;

}

int prcd(symbol)

{

switch(symbol)

{

case '+':

case '-':

return 2;

break;

case '\*':

case '/':

return 4;

break;

case '$':

case '^':

return 6;

break;

case '#':

case '(':

case ')':

return 1;

break;

}

}

int op(char symbol)

{

switch(symbol)

{

case '+':

case '-':

case '\*':

case '/':

case '^':

case '$':

case '&':

case '(':

case ')':

return 1;

break;

default:

return 0;

}

}