13. Problem Statement: Infix to Postfix

**Problem Analysis:**

There is an algorithm to convert an infix expression into a postfix expression. It uses a stack; but in this case, the stack is used to hold operators rather than numbers. The purpose of the stack is to reverse the order of the operators in the expression. It also serves as a storage structure, since no operator can be printed until both of its operands have appeared.

In this algorithm, all operands are printed (or sent to output) when they are read. There are more complicated rules to handle operators and parentheses.

**Algorithm:**

*1.* Scan the infix expression from left to right.

*2.* If the scanned character is an operand, output it.

*3.* Else,

*3.1* If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty), push it.

*3.2* Else, Pop the operator from the stack until the precedence of the scanned operator is less-equal to the precedence of the operator residing on the top of the stack. Push the scanned operator to the stack.

*4.* If the scanned character is an ‘(’, push it to the stack.

*5.* If the scanned character is an ‘)’, pop and output from the stack until an ‘(’ is encountered.

*6.* Repeat steps 2-6 until infix expression is scanned.

*7.* Pop and output from the stack until it is not empty.

**Source Code:**

#include<stdio.h>

char stack[20];

int top = -1;

void push(char x) {

stack[++top] = x; }

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

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

return 1;

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

return 2;

}

int main()

{

char exp[20];

char \*e, x;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

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

{

if(isalnum(\*e))

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

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

push(\*e);

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

{

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

printf("%c", x);

}

else

{

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

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

push(\*e);

}

e++;

}

while(top != -1)

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

}

**Sample Input:**

Enter the expression :: (a+b)\*c+(d-a)

**Sample Output:**

ab+c\*da-+