Converting An Infix Expression To Postfix

0	1	2	3	4	5
'A'	'+'	'B'	'*'	'C'	'\0'

Ele	Stack	Postfix
'A'	empty	A
'+'	'+'	A
'B'	'+'	AB
'*'	'+', '*'	AB
'C'	'+', '*'	ABC
end	empty	ABC*+

A	$	B	/	C	-	D	*	E	/	F	+	G	'\0'

AB$C/DE*F/-G+

Ele	Stack	Postfix
A		A
$	$	A
B	$	AB
/	/	AB$
C	/	AB$C
-	-	AB$C/
D	-	AB$C/D
*	-, *	AB$C/D
E	-, *	AB$C/DE
/	-, /	AB$C/DE*
F	-, /	AB$C/DE*F
+	+	AB$C/DE*F/-
G	+	AB$C/DE*F/-G
end	empty	AB$C/DE*F/-G+

A * B $ C - E + F / G % H * I

Ele	Stack	Postfix
A		A
*	*	A
B	*	AB
$	*, $	AB
C	*, $	ABC
-	-	ABC$*
E	-	ABC$*E
+	+	ABC$*E-
F	+	ABC$*E-F
/	+, /	ABC$*E-F
G	+, /	ABC$*E-FG
%	+, %	ABC$*E-FG/
H	+, %	ABC$*E-FG/H
*	+, *	ABC$*E-FG/H%
I	+, *	ABC$*E-FG/H%I
end	empty	ABC$*E-FG/H%I*+

Algorithm For Converting Infix To Postfix
================================

1. Scan the given INFIX expression from left to right , one character at a time.

2. Check whether it is an operand or operator.

3. If it is an OPERAND , then copy it in the POSTFIX array and goto step 5.

4. If it is an OPERATOR then:

    a. Check whether the STACK is empty or not.
    b. If STACK is empty , then PUSH the operator in it and goto step 5
    c. If STACK is not empty then:
        i. Check the PRECEDENCE of OUTSIDE operator with STACKTOP.
        ii. If OUTSIDE operator is greater then PUSH it and goto step 5
        iii. If the precedence of STACKTOP is higher or equal then POP it, copy it in the POSTFIX array and goto step 4a

5. Repeat the above steps until INFIX expression ends.

6. Finally , POP the remaining operator from the STACK and COPY them in the POSTFIX array.

7. Finish and return

Implementing Conversion Of Infix To Postfix

```
==================================
struct Stack
{
    char arr[10];
    int tos;
};
void push(struct Stack *,char);
char pop(struct Stack *);
int isoperand(char);
int isempty(struct Stack);
int prcd(char,char);
void convert(char [ ],char [ ]);
int main()
{
   // write code of main() function
}
```