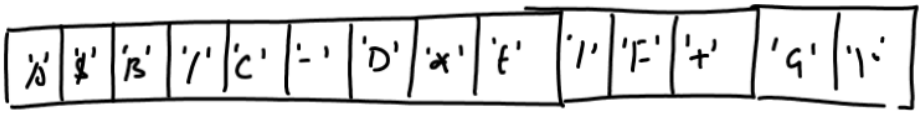
Remaining Functions Of Converting Infix To Postfix

```
void push(struct Stack *p,char x)
{
   if(p->tos==MAX-1)
   {
      printf("Stack Overflow");
      return;
   }
   p->tos++;
   p->arr[p->tos]=x;
}
char pop(struct Stack *p)
{
   char x;
   if(p->tos==-1)
   {
      printf("Stack Underflow");
      return 0;
   }
   x=p->arr[p->tos];
   p->tos--;
   return x;
}
```

```
int isoperand(char ch)
{
   if((ch>='A' && ch<='Z')||(ch>='a' && ch<='z')||(ch>='
      return 1;
   return 0;
}
int isempty(struct Stack S)
{
   return (S.tos==-1);
}
int prcd(char op1,char op2)
{
   if(op2=='$')
      return 0;
   else if(op1=='$')
      return 1;
   else if(op2=='/'||op2=='*'||op2=='%')
      return 0;
   else if(op1=='/'||op1=='*'||op1=='%')
      return 1;
   else if(op2=='+'||op2=='-')
      return 0;
   else
      return 1;
```

+-/$ABC/*DEFG

Converting Infix To Prefix



Ele	Stack	Prefix
'G'		G
'+'	'+'	G
'F'	'+'	GF
'/'	'+','/'	GF
'E'	'+','/'	GFE
'*'	'+','/','*'	GFE
'D'	'+','/','*'	GFED

Ele	Stack	Prefix
'-'	'+','-'	GFED*/
'C'	'+','-'	GFED*/C
'/'	'+','-','/'	GFED*/C
'B'	'+','-','/'	GFED*/CB
'$'	'+','-','/','$'	GFED*/CB
'A'	'+','-','/','$'	GFED*/CBA
end		GFED*/CBA$/-

Algorithm For Converting Infix To Prefix
==========

1. Scan the given INFIX expression from RIGHT to LEFT one character at a time.

2. Check whether it is an OPERAND or OPERATOR.

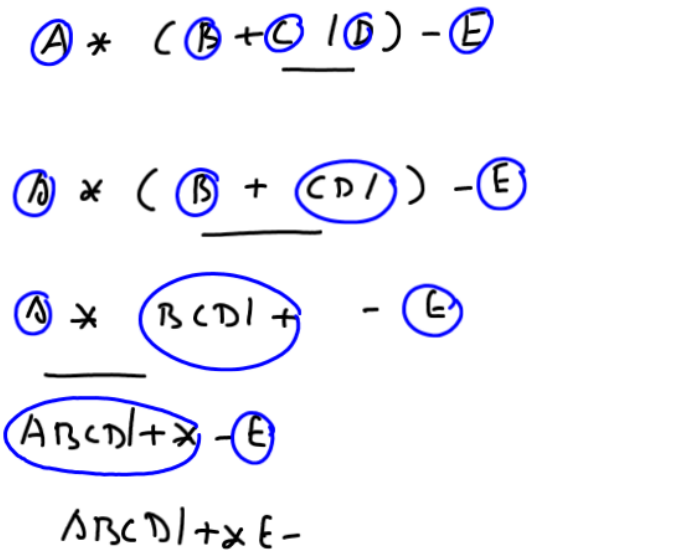
3. If it is an OPERAND then copy it in the PREFIX array and goto step 5

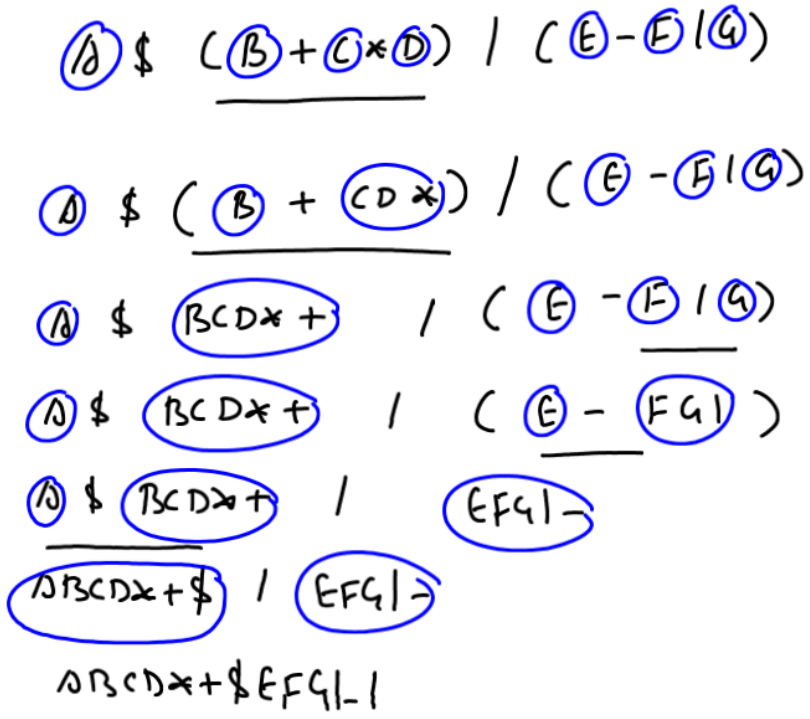
4. If it is an operator then :

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

5. Repeat the above process until the INFIX expression ends.
6. POP the remaining operators from the STACK and copy them in PREFIX array
7. Reverse the PREFIX array
8. Finish and return

Converting Parenthesized Infix To Postfix
==========





A * (B + C / D) - E

Ele	Stack	Postfix
A		A
*	*	A
(	*, (	A
B	*, (	AB
+	*, (, +	AB
C	*, (, +	ABC
/	*, (, +, /	ABC
D	*, (, +, /	ABCD
)	*	ABCD/+
-	-	ABCD/+*
E	-	ABCD/+*E
end	empty	ABCD/+*E-

A $ (B + C * D) / (E - F / G)

Ele	Stack	Postfix
A		A
$	$	A
(	$, (	A
B	$, (	AB
+	$, (, +	AB
C	$, (, +	ABC
*	$, (, +, *	ABC
D	$, (, +, *	ABCD
)	$	ABCD*+
/	/	ABCD*+$
(	/, (	ABCD*+$
E	/, (	ABCD*+$E
-	/, (, -	ABCD*+$E
F	/, (, -	ABCD*+$EF
/	/, (, -, /	ABCD*+$EF
G	/, (, -, /	ABCD*+$EFG
)	/	ABCD*+$EFG/-
end	empty	ABCD*+$EFG/-

Algorthm For Converting Parenthesized Infix To Postfix
==========

1. Scan the given INFIX expression left to right one character at a time and proceed in usual way until an opening bracket arrives.

2. As soon as an opening bracket arrives , PUSH it in the STACK.

3. Also PUSH the first operator whcih arrives after the opening bracket.

4. Again proceed normally until a closing bracket arrives.

5. As soon as a closing bracket arrives do the following:

    a. POP all the operators from the STACK uptill opening bracket.
    b. Copy all the operators in the POSTFIX array except opening bracket.
    c. Discard both the brackets, means , neither opening bracket should be copied in the POSTFIX array nor the closing bracket should be PUSHED in the STACK.

6. Again proceed normally and if an opening bracket arrives again then goto step 2.

7. Repeat the above process until the INFIX expression ends.

8. POP the remaining operators from the STACK and copy them in the POSTFIX array one by one.