A + B * C / D $ E

A + B * C / D E $

A + B C * / D E $

A + B C * D E $ /

A B C * D E $ / +

infix	A	+	B	*	C	/	D	$	E	\0
	0	1	2	3	4	5	6	7	8	9

Char	Stack	Postfix
'A'		A
+	+	A
B	+,	AB
*	+,*	AB
C	+,*	ABC
/	+,/	ABC*
D	+,/	ABC*D
$	+,/,$	ABC*D
E	+,/,$	ABC*DE
\0		

ABC*DE$/+

```
struct stack
{ char st[10];
  int tos;
};
void push(struct stack*, char); ✓
char pop(struct stack*); ✓
int isempty(struct stack);
int isoperand(char);
int preced(char, char);
void convert(char[], char[]);
```

```
void main()
{
   char infix[40], postfix[40];
  printf("Enter the infix expression=");
 scanf("%s", infix);
 convert(infix, postfix);
 printf("postfix expression is=%s", postfix);
 getch();
                                        ABC*+
}
```

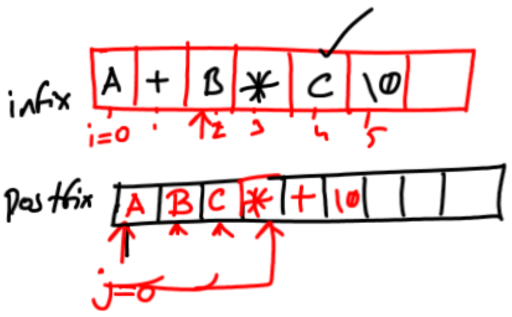
```c
void push ( Struct Stack *P , char c)
{ if ( p->tos = = 9)
  {     printf(" Stack is overflow ");
        return ;
  }
   P->tos = P->tos+1;
  P->st[p->tos]= c ;
}
char pop( struct stack *p)
{   char c;
    if( p->tos= = -1)
    {     printf(" Stack is a underflow ");
    }      return (-1);
```

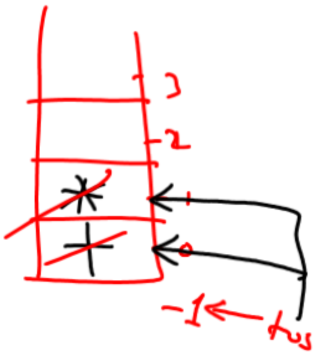
```c
c=p->st [p->tos];
 p->tos--;
 return (c);
}

int isempty (struct stack s)
{ if(s.tos= = -1)
  else  return 1;
          return 0;
  }
```

```c
void  Convert ( char infix[], char posfix[])
{     int  i, j=0, result;
      Struct Stack  s;
      char  c;
      s.tos = -1;
  for( i=0; infix[i] != '\0' ; i++)
  {       c = infix[i];
      if( isoperand(c)= = 1)
      {     postfix[j]= c;
             j++;
      }
      else
      {
            while( isempty(s)= =0)
            {
              result=pred( c , s.st[s.tos]);
```

```c
if(result = = 0)
{  posfix[j] = pop(&s);
    j++;
}
else
     break;
}   // while end
Push( &s , c );
}  // else end
}  // for end
```

```
while ( isempty(s) = = 0)
{     postfix[j]= pop (&s);
       j++;
}
postfix [j] = '\0';

}
```





```
int  isoperand ( char  ch)
{
 if( (ch>='A' && ch<='Z') || (ch>='a' && ch<='z') ||
       (ch >= '0' && ch<='9'))
         return (1);
else
 return 0;
}
int  precd(char op1 , char  op2)
{
     if( op2 = = '$')
          return 0;
     else  if( op1 = = '$')
          return (1);
     else if( op2=='/' ||
int precd( 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;
}
```