converting from infix to postfix

▨ Conditions ▨

- In stack precedence and incoming precedence are as follows.

/* isp and icp arrays – index is value of precedence
lparen, rparen, plus, minus, times, divide, mod, eos */
static int isp[] = { 0, 19, 12, 12, 13, 13, 13, 0 };
static int icp[] = { 20, 19, 12, 12, 13, 13, 13, 0 };

- Scan infix expression from left to right

```
void postfix(void)
{/* output the postfix of the expression. The expression
    string, the stack, and top are global */
  char symbol;
  precedence token;
  int n = 0;
  top = 0;    /* place eos on stack */
  stack[0] = eos;
  for (token = getToken(&symbol, &n); token != eos;
                          token = getToken(&symbol,&n)) {
    if (token == operand)
      printf("%c", symbol);
    else if (token == rparen) {
      /* unstack tokens until left parenthesis */
      while (stack[top] != lparen)
        printToken(pop());
      pop();  /* discard the left parenthesis */
    }
    else {// operator, lparen
      /* remove and print symbols whose isp is greater
         than or equal to the current token's icp */
      while(isp[stack[top]] >= icp[token])
        printToken(pop());
      push(token);
    }
  }
  while ( (token = pop()) != eos)
    printToken(token);
  printf("\n");
}
```

Input string : ((a+b)+c*d+e)/a

Token	Stack [0] [1] [2] [3] [4]	Top	Output
	eos	0	
(	eos (	1	
(	eos ((	2	
a	eos ((	2	a
+	eos ((+	3	a
b	eos ((+	3	ab
)	eos (	1	ab+
+	eos (+	2	ab+
c	eos (+	2	ab+c
*	eos (+ *	3	ab+c
d	eos (+ *	3	ab+cd
+	eos (+	2	ab+cd*+
e	eos (+	2	ab+cd*+e
)	eos	0	ab+cd*+e+
/	eos /	1	ab+cd*+e+
a	eos /	1	ab+cd*+e+a
eos		-1	ab+cd*+e+a/