武聖夜市	三、六	4	台南市
民雄夜市	一	2	嘉義縣
東石夜市	二	3	嘉義縣
士林夜市	每日	3	台北市
公館夜市	每日	3	台北市
師大夜市	每日	4	台北市
逢甲夜市	每日	3	台中市
大甲夜市	四	2	台中市

Note: You must use an array and struct; otherwise, no points will be given.

Test Case

Please test your program with Input and, and then check the answers with Output.

Listing 5: Structure and array

```
Input：
花園夜市,467,5,台南市
大東夜市,125,4,台南市
武聖夜市,36,4,台南市
民雄夜市,1,2,嘉義縣
東石夜市,2,3,嘉義縣
士林夜市,1234567,3,台北市
公館夜市,123567,3,台北市
師大夜市,1234567,4,台北市
逢甲夜市,1234567,3,台中市
大甲夜市,4,2,台中市
Output：
評價最高的夜市：花園夜市
星期三有開的夜市：武聖夜市、士林夜市、公館夜市、師大夜市、逢甲夜市
台中的夜市：逢甲夜市、大甲夜市
```

6. Infix to Postfix Conversion

Please finish a program which can change the expression from the infix form to the postfix form in C using a stack. The input operands are lowercase letters「*a*」to「*z*」; meanwhile, the input operators are 「*」,「/」,「+」, and 「-」. Furthermore, the precedence of the operators is「*」=「/」>「+」 =「-」. The output includes two lines of the results: (1) the postfix expression, and (2) the maximum number of elements in the stack during the computation.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 6: Infix to Postfix Conversion.

```
Input1 ：
a + b * c
Output1 ：
a b c * +
2

Input2 ：
a / b - c + d * e - a * c
Output2 ：
a b / c - d e * + a c * -
2
```

7. Postfix expression evaluation

Please finish a program which can perform postfix expression evaluation and count the max top in stack during the process in C. The input is a prefix expression. The operands are lowercase letters 「*a*」 to 「*z*」; the input operators are 「*」, 「/」, 「+」, and 「-」. Furthermore, the precedence of the operators is 「*」 = 「/」 > 「+」 = 「-」. First, convert the prefix expression to the postfix form. After that, you should use a stack to process postfix form and record the stack size for each step. The output includes two results: (1) the postfix expression, and (2) the stack size for each step.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 7 : Postfix expression evaluation

```
Input ：
- + a * - / b c d e / f g

Output ：
a b c / d - e * + f g / -
```