A - Scanner Test

請注意不能用 #include<regex> 抓到 BA 0 分!!! 請注意不能用 #include<regex> 抓到 BA 0 分!!! 請注意不能用 #include<regex> 抓到 BA 0 分!!!

Description

在編譯器中,Token 是組成程式碼的最小單位,需要由 Scanner 先把 Input text 轉換 Token,Parser 才會進行後續的工作。請利用表一,撰寫一個 Scanner 來取得 Token 並輸出。

表一

Terminal	Regular Expression
NUM	[1-9][0-9]* 0
ID	[A-Za-z]+
OP	[\+\-*/=]
LPR	\(
RPR	\)

Input Format

輸入一行"運算式,每筆測試資料只會有一個運算式,但其中可能會來雜 多個空格。

部分輸入的程式碼將會有不符合 Regular Expression 的情況發生。

Output Format

請在切割後輸出其 Token 種類。

若為數字,需附上其數值,並以一個空白做為區隔。

若為運算子,需附上其符號,並以一個空白做為區隔。

例如 0 則需輸出 NUM 0,以此類推

例如 + 則需輸出 OP + , 以此類推

若不符合 Regular Expression,即使只出現一次錯誤,則只需印出"invalid input",前面合法輸出則不印出。

每個 token 輸出後請以\n 分隔。

int a = 10 D int D a OP = NUM 10	Sample input 1	Sample output 1
OP = NUM 10	int a = 10	ID int
NUM 10		ID a
Sample input 2		OP =
A(10 + 11 + 12) ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR Sample input 3 "int Sample output 4 invalid output Sample output 4 A(10+11+12)\$ Sample output 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 11 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		NUM 10
A(10 + 11 + 12) ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR Sample input 3 "int Sample output 4 invalid output Sample output 4 A(10+11+12)\$ Sample output 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 11 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		
LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR Sample input 3 "int invalid output 4 A(10+11+12)\$ Sample output 4 A(10+11+12)\$ Sample input 5 K (A(10 + 11 + 12) * Hello (999)) LPR NUM 10 OP + NUM 10 OP + NUM 11 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR	Sample input 2	Sample output 2
NUM 10	A(10 + 11 + 12)	ID A
OP + NUM 11 OP + NUM 12 RPR Sample input 3 "int invalid output 4 A(10+11+12)\$ Sample output 4 invalid output Sample input 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		LPR
NUM 11		NUM 10
OP + NUM 12 RPR Sample input 3 "int invalid output 4 A(10+11+12)\$ Sample output 4 A(10+11+12)\$ Sample output 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		OP +
NUM 12 RPR		NUM 11
RPR Sample input 3 Sample output 4 invalid output		OP +
Sample input 3		NUM 12
"int invalid output Sample input 4 A(10+11+12)\$ invalid output Sample input 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		RPR
Sample input 4 A(10+11+12)\$ invalid output Sample output 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR	Sample input 3	Sample output 4
A(10+11+12)\$ invalid output	"int	invalid output
Sample input 5 K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR	Sample input 4	Sample output 4
K (A(10 + 11 + 12) * Hello (999)) ID K LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR	A(10+11+12)\$	invalid output
LPR ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR	Sample input 5	Sample output 5
ID A LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR	K (A(10 + 11 + 12) * Hello (999))	ID K
LPR NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		LPR
NUM 10 OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		ID A
OP + NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		LPR
NUM 11 OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		NUM 10
OP + NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		OP +
NUM 12 RPR OP * ID Hello LPR NUM 999 RPR		NUM 11
RPR OP * ID Hello LPR NUM 999 RPR		OP +
OP * ID Hello LPR NUM 999 RPR		NUM 12
ID Hello LPR NUM 999 RPR		RPR
LPR NUM 999 RPR		OP *
NUM 999 RPR		ID Hello
RPR		LPR
		NUM 999
RPR		RPR
<u> </u>		RPR