# C definition

## Description

> "I want to write my own compiler!" says NightA.
>
> But soon after that, he found that he cannot figure out how to implement the macro preprocessor. And he stuck.

So, to help NightA finish his dream, you are required to write a **macro preprocessor**, that take in a stream of codes containing normal statements and `#define` and `#undef` statement, output the processed code with macro expansions and do not output the macro statements.

To reduce your work, NightA kindly added whitespace all over the place, including both sides of variable names, operands, signs. Function names will  have whitespace on the left side, left parentheses (`(`) on the right side.

Since the whitespace is not that important, NightA will use SPJ on this problem. All whitespaces like ` `, `\t`, `\n`... will be ignored.

And here is the details:

1. If `#define a b` appears, then all the single `a` after this statement will be replaced by `b`.
2. If `#undef a` appears, then the replacement rule about `a` will be removed.

These are corresponding to **Sample1**.

3. If `#define mul( a , b ) a * b` appears, then all the `mul( x , y )` will be replaced by `x * y`. The difference between two function macros will only count function name and numbers of parameters. That means `#define mul( a , b ) xx` is equivalent to `#define mul( c , d ) xx`. The input will guarantee that if a `(` appears, then there must be a non-empty parameter list.
4. If two function macros have the same function name but different number of parameters, then they are two different defines.

This is corresponding to **Sample2**.

5. For simplicity, you are not required to handle macros in macro.

This is corresponding to **Sample3**.

Inputs guarantee that `#` will only appear at the beginning of the line. And only have `#define` & `#undef`

If undef an element that have not been defined, output `ERROR NFOUND` and stop, ignore all the inputs behind.

If define an element that have been defined, output `ERROR DUP` and stop, ignore all the inputs behind.

Inputs guarantee that all the other inputs are legal.

## Sample

## Sample1

Input

```
#define aa bb
aa bb cc
#undef aa
aa
```

Output

```
bb bb cc
aa
```

## Sample2

Input

```
#define mul c
#define mul( a , b ) a * b
#define mul( a ) a * a
mul( 123 + 456 , 789 )
mul( hihi )
mul
#undef mul( b )
mul( hihi )
```

Output

```
123 + 456 * 789
hihi * hihi
c
mul( hihi )
```

## Sample3

Input

```
#define aa bb
#define cc aa
cc
```

Output

```
aa
```

## Hint

Again, this problem will use SPJ. All whitespaces like ` `, `\t`, `\n`... will be ignored, no matter where they are.

The input will not exceed 1000 lines. The number of parameters will not exceed 100. One variable/function name's length will not exceed 100. Total characters will not exceed 10000.

10pts: Input will have `#define` and `#undef` with no parameter, but no statement will be replaced.

20pts: Defines will not have parameters.

70pts: No specific rules.