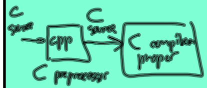
Recall: proposed regexp for comments

```
"/*"([^*]|"*"[^/])*"*/"
```

doesn't match ("*"?)

```
/***/
/* -------- ****/
/* ---- */ ----*/
```

HW1:




```
#include "mydefs.h"
#define NO  0&*whoops
#define YES 1
    :
    if (NO) { .... }
         0

#define FOO BAR
```

In general:

#define <key> <value> \n

(w.s. not \n before key; key is id; w.s. (space or tab)+ before value)

```
#define expr (FOO || BAR)\
    && BAT
```

Example

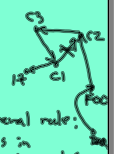
```
#define FOO BAR
#define BAR 5
int x = FOO;
#define BAR 6   <-
int y = FOO;
```

```
#define BAR 5
#define FOO BAR
x = FOO;
    5
#define BAR 5
#define BAR 6
y = FOO;
```

---

cycles

key o——> o value




```
#define C1  C2
#define C2  C3
#define C3  C1
#define FOO C2
int x = FOO;
        C2
int y = C1;
        C1
int z = C3;
        C3
```

General rule: keys in key-value cycles are treated as nonkeys

```
/*
But when a cycle appears
you can't remove it! */
#define C1 17
int a = FOO;
        17

FOO -> C2 -> C3 -> C1 -> C2
int y = BAR;
        C2
```

Look for new/broken cycles when the dictionary is altered (when processing a define directive). Include a flag with each key indicating whether or not it is in a cycle.

Example: Assume FOO is not already in a cycle

```
#define FOO BAR
```

follow chain forward from BAR until

1) get a nonkey
   — no cycle, so done
2) get FOO — cycle!
   traverse cycle again setting flags on the whole cycle
3) get key whose "in cycle" flag is on. — same as 1.

But first! check if definition breaks an existing cycle. Flag of key ("FOO") is on. If so, run through existing cycle, turning flags off.

Jan 19-10:55 AM

Summary:
when processing a new
#define directive: (flag)
1) check if key is in a cycle
If so, traverse cycle, turning
flags off.
2) Add (or change) the link
from key (to the new
value)
3) Check for new cycle:
newvalue → ... → key
if so, set all the flags.

Saving a string in flex:

```
%{
#include <stdlib.h>
char *stash_string(char *str);
%}

id      [_A-Za-z][_A-Za-z0-9]*

%%
...
{id}    { char *newid = stash_string(yytext);
          ...
        }
...
%%
char *stash_string(char *str)
{
  char *buf = (char *)malloc(
                strlen(str) + 1);
  strcpy(buf, str);
  return buf;
}

int strcmp(char *s1, char *s2)
  returns -1 if s1 < s2
           0 if s1 == s2
           1 if s1 > s2
if (!strcmp(s1,s2))
    if s1 & s2 are identical
```

Dictionary?
Use a linked-list hash table

```
typedef struct n {
  int in_cycle; /* boolean flag */
  char *key;
  char *value;
  struct n *next;
} NODE_REC, *node;
```

Jan 19-11:51 AM