

```
Homework 1
How ($) hoc works

#define MAX 10 value
...
if (i < max) → (i < 10)

#define (max) value
id      ops or id
        char *
→ ' ' ' " &

#define MAX 10
i < MAX
    ^
#define MAX 50
i < max (old association is lost)

#define FOO S
#define BAR FOO
#define BAR FOO
int i = BAR;
         S
#define BAR BAT
int BAR=FOO;
int BAT=S;
printf("%d", BAR);
```

Jan 26-10:59 AM

```
#define TOO BAR
#define BAR  $\frac{mid}{5}$ 

int a[ $\frac{foo}{5}$ ];

#define FOO 6
int b[ $\frac{foo}{5}$ ];
int c[ $\frac{bar}{5}$ ];

int c[ $\frac{mid}{5}$ ];

int  $\frac{bar}{5} = 5$ ;

#define CY CY
int  $\frac{cy}{5} = 0$ ;
```

Directed graph
vertices are keys and values

key $\longrightarrow$ value

if $\frac{key}{5}$ $\neq$ $\frac{value}{5}$ then key value is in effect.

Jan 26-11:18 AM

```
#define CY1 CY2
#define CY2 CY3
#define CY3 CY1

#define F00 CY2

int F00 = 10;
CY2
int CY3 = 5;
CY3
```

#define CY1 HELLO
double F00 = 0.3;
HELLO

Look for a cycle when a new link is added to the graph (dictionary).

add

Jan 26-11:28 AM

hashing

% { index distribution
for pointers
to objects
strings, etc.
objects }

%.} key + flag.

names bound to reg ops

defo

"register"
{index
{str op}}

(A)

flag =
key
→ key value
flag = 0
next direction

Dictionary

Set of key-value pair
search for value associated with a key

add a new pair
update a key to a new value
mark keys as being on a cycle

un-mark

101 00000000

Jan 26-11:45 AM

main

key

value

ptr

union

```

struct foo {
    int f1;
    char *f2;
    double f3;
} *ptr;

```

ptr

int

char *

double

$ptr \rightarrow f3 = 3.14;$

union

```

foo {
    int f1;
    char *f2;
    double f3;
} *ptr;

```

ptr

int

char *

double

Jan 26-12:06 PM