clicker.cs.illinois.edu

Q1

~Code~
340

CS 340

Exam 1 Review

Updates

1. Exam 1 - This Thursday
    a. **Sign up now!**
    b. **Study guide & Practice Exam out**
    c. **No class Thursday**

1. HW3 - Due tomorrow night at midnight

1. MP3 - Due Tuesday next week (Feb 24th)

Agenda

1. Review and Questions

1. HW 2 Coding Review

1. Practice Exam Coding Review

How I made the Exam...

* see study guide for details

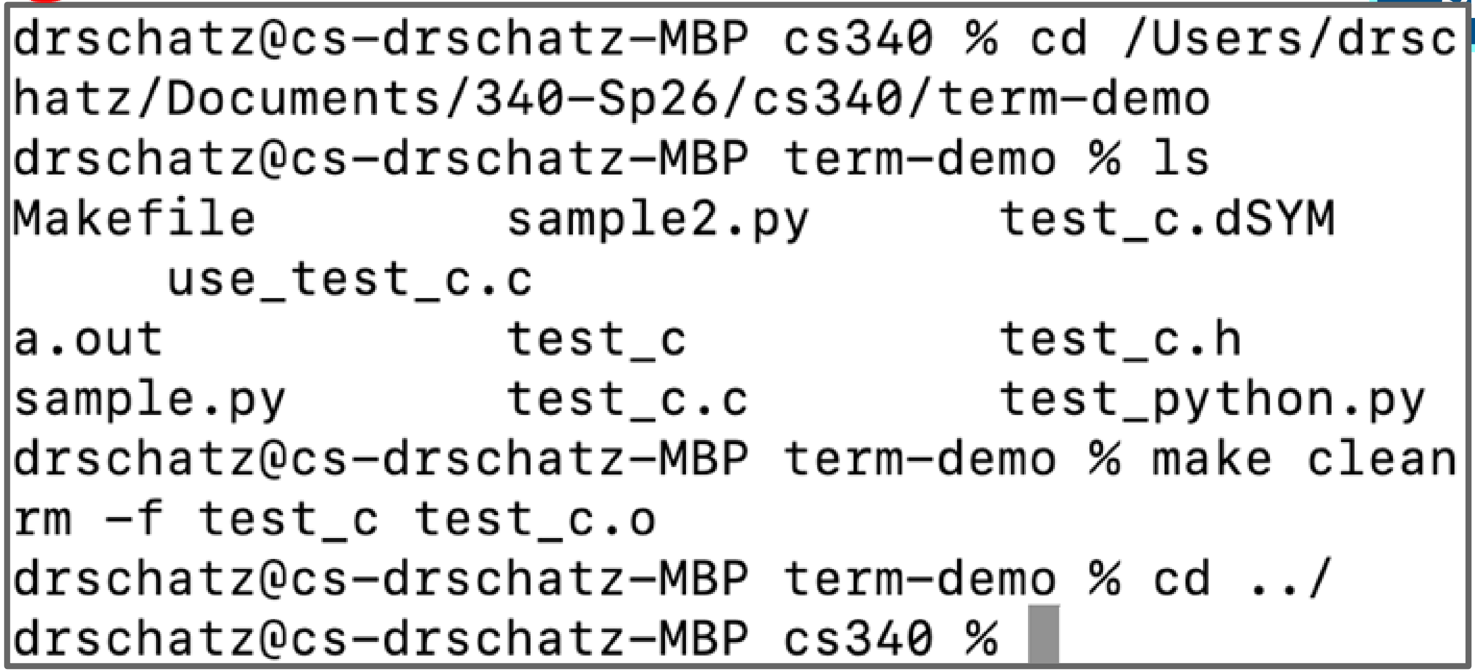
How would I compile use_test_c.c and test_c.c?

gcc term-demo/test_c.c term-demo/use_test_c.c

clicker.cs.illinois.edu

Q2

~Code~ 340



```
drschatz@cs-drschatz-MBP cs340 % cd /Users/drsc
hatz/Documents/340-Sp26/cs340/term-demo
drschatz@cs-drschatz-MBP term-demo % ls
Makefile                sample2.py              test_c.dSYM
     use_test_c.c
a.out                   test_c                  test_c.h
sample.py               test_c.c                test_python.py
drschatz@cs-drschatz-MBP term-demo % make clean
rm -f test_c test_c.o
drschatz@cs-drschatz-MBP term-demo % cd ../
drschatz@cs-drschatz-MBP cs340 % 
```

Follow up, which is true?

clicker.cs.illinois.edu

Q3

~Code~
340

could have different ISAs

✗ ~~A) The executable created will work on any computer that can run C.~~

✓ B) The C code will compile on any computer that can compile C.

✓ C) If I make a change to test_c.c I need to recompile the code again. ← to see the change

✓ D) I can see the executable created in my file finder window after creating it in the terminal.

✓ E) The executable can be interpreted by my CPU.

Big Ideas #1

ls, pwd, cd

Compiling code creates an executable per your ISA

C is portable but needs to be compiled first

Machine code is not always portable ~~but can~~

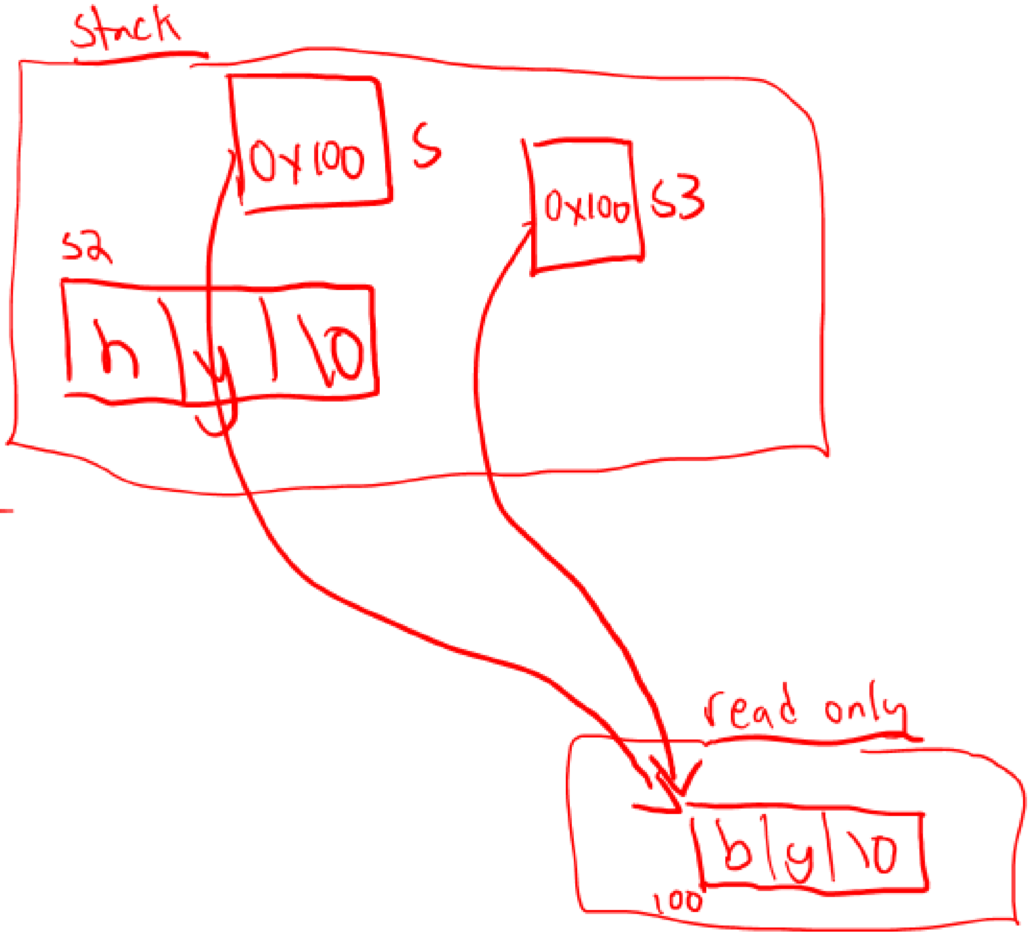
C-Strings -bytes need to live somewhere

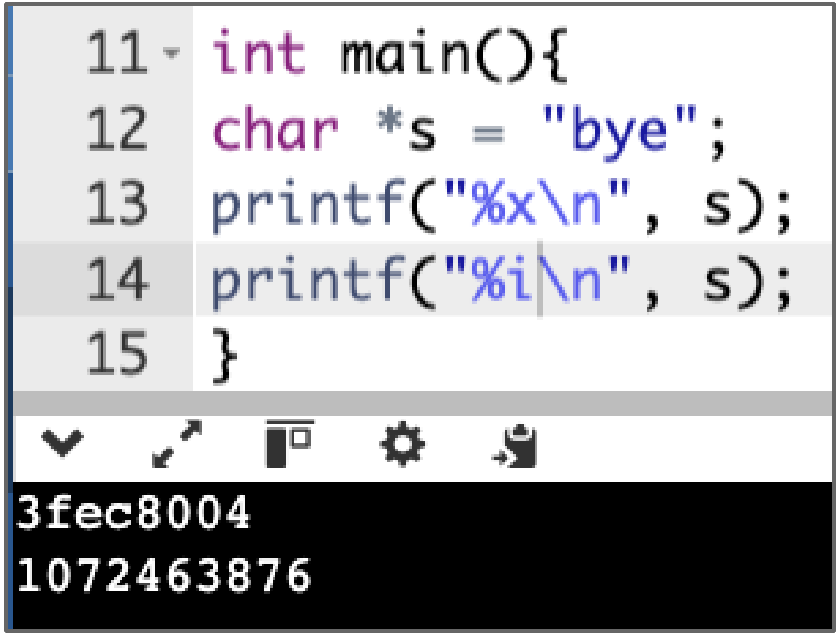
```
char *s = "by";

char s2[3] = "hy";
    char *s3 = s2;
```

~~s[0] = 'x';~~ ✗ does not work

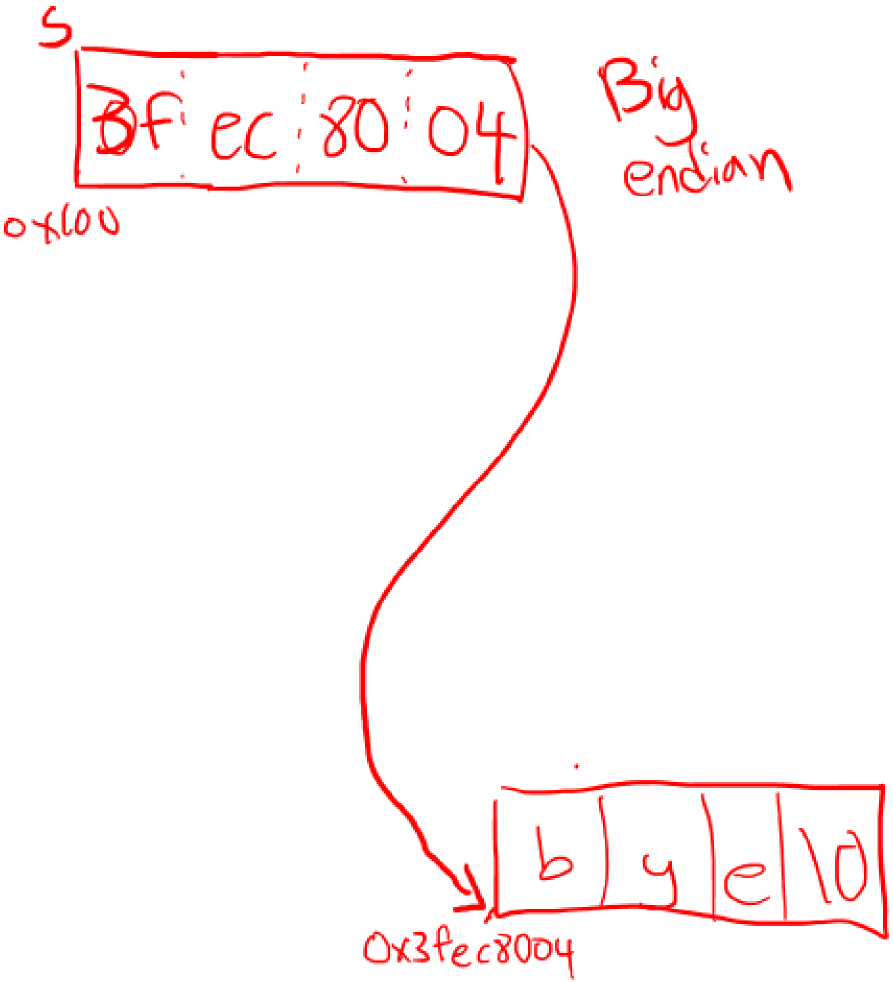
s2[0] = 'x'; ✓ does work





```c
int main(){
char *s = "bye";
printf("%x\n", s);
printf("%i\n", s);
}
```

```
3fec8004
1072463876
```

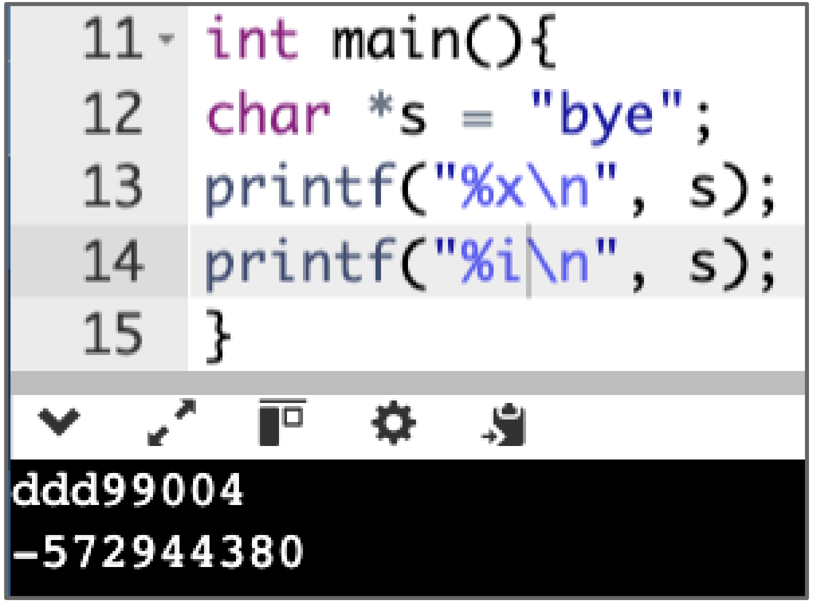


You computer uses little endian... what do the bytes of s look like?

clicker.cs.illinois.edu

Q4

~Code~
340



```c
int main(){
char *s = "bye";
printf("%x\n", s);
printf("%i\n", s);
}
```

```
ddd99004
-572944380
```

A) dd d9 90 04

B) 57 29 44 38

[✓ C) 04 90 d9 dd]

D) 38 44 29 57

E) 40 09 9d dd

F) 83 44 92 75

Why is everything stored as a 1 or 0 in a computer?

clicker.cs.illinois.edu

Q5

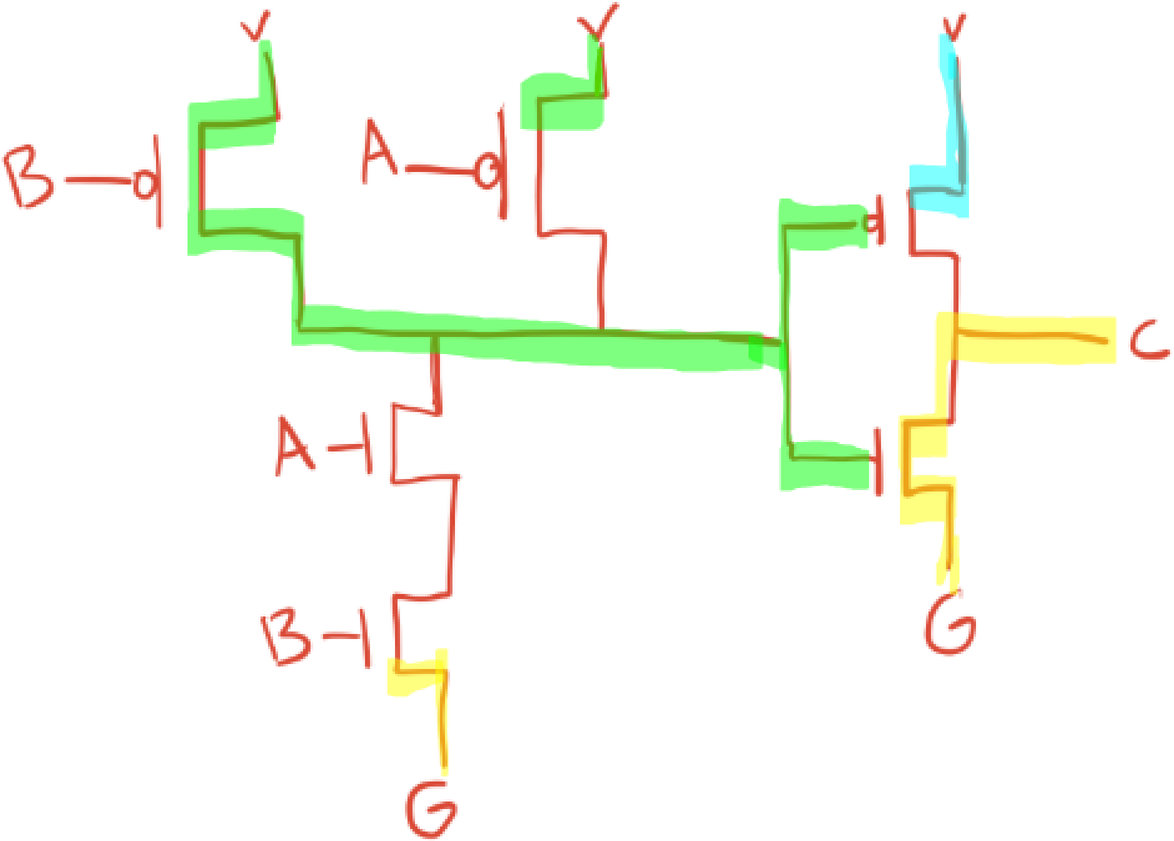
~Code~
340

[✓ A)] The hardware is built to recognize and use 2 values of voltage.

~~B) Binary is easier to work with then decimal.~~ ← only because A

~~C) Caching only works with binary.~~

~~D) Bytes are 1's and 0's.~~



A=1 B=0

C = 0

What can we build with transistors?

clicker.cs.illinois.edu

Q6

~Code~
340

~~A) Anything we can imagine~~

[✓ B)] Any computational logic including selection and math

~~C) Any computational logic including math but not selection~~

~~D) Only NAND NOT XOR logic~~

selection
= MUX

What is the minimum depth of 2-MUX's needed for a 9 selection?

clicker.cs.illinois.edu

Q7

~Code~
340

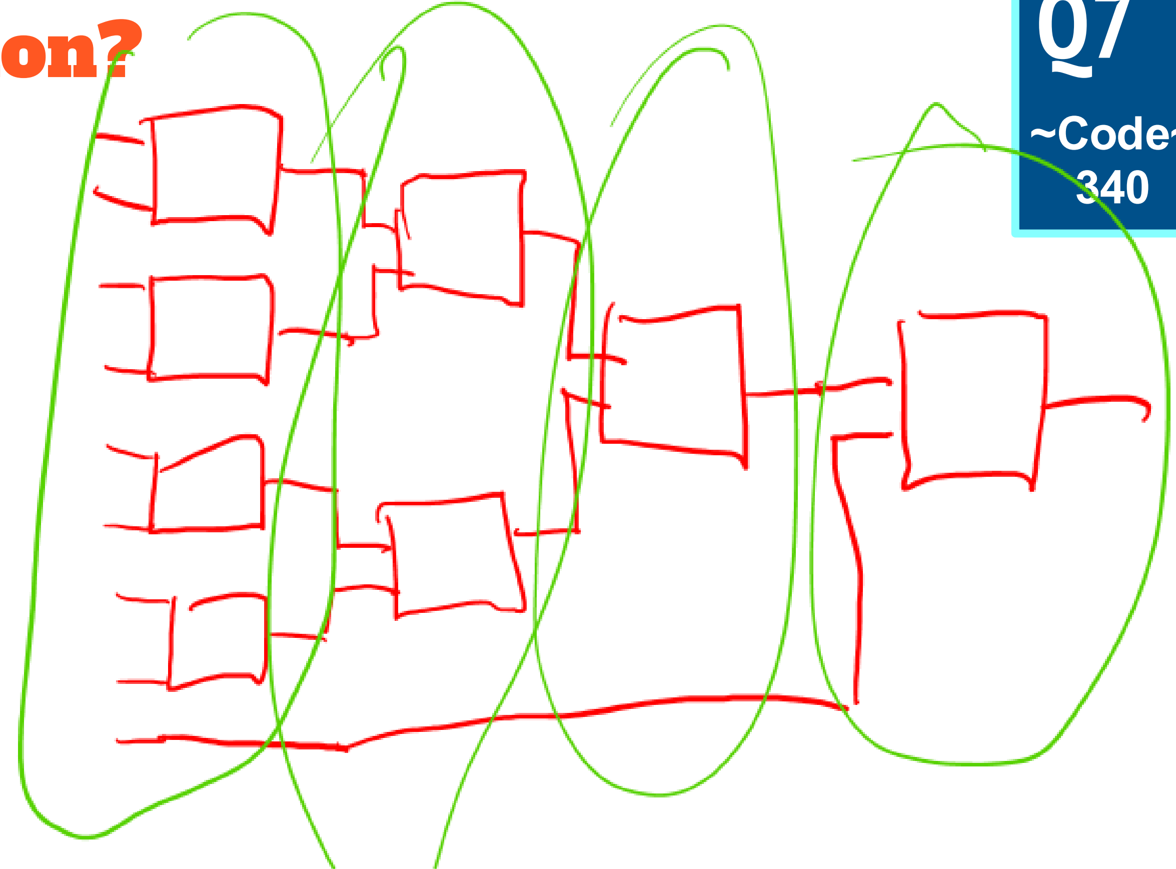
A) 1

B) 2

C) 3

[✓ D) 4]

E) 5



Hardware for Storing Information

registers - fast, expensive

Memory -
RAM

Disk - slow, cheap

registers in the cpu

RAM for memory

Disk for persistent
all data

caching

What is true about caching?

it would just be slow

A) ~~You cannot build a computer without caching~~.

[✓ B)] You computer would be slow without caching

C) ~~The best caching algorithms puts a little bit of every program in the top layer of a cache for quick access.~~

locality

D) ~~Registers are used in caching.~~

CPU

clicker.cs.illinois.edu

Q8

~Code~
340

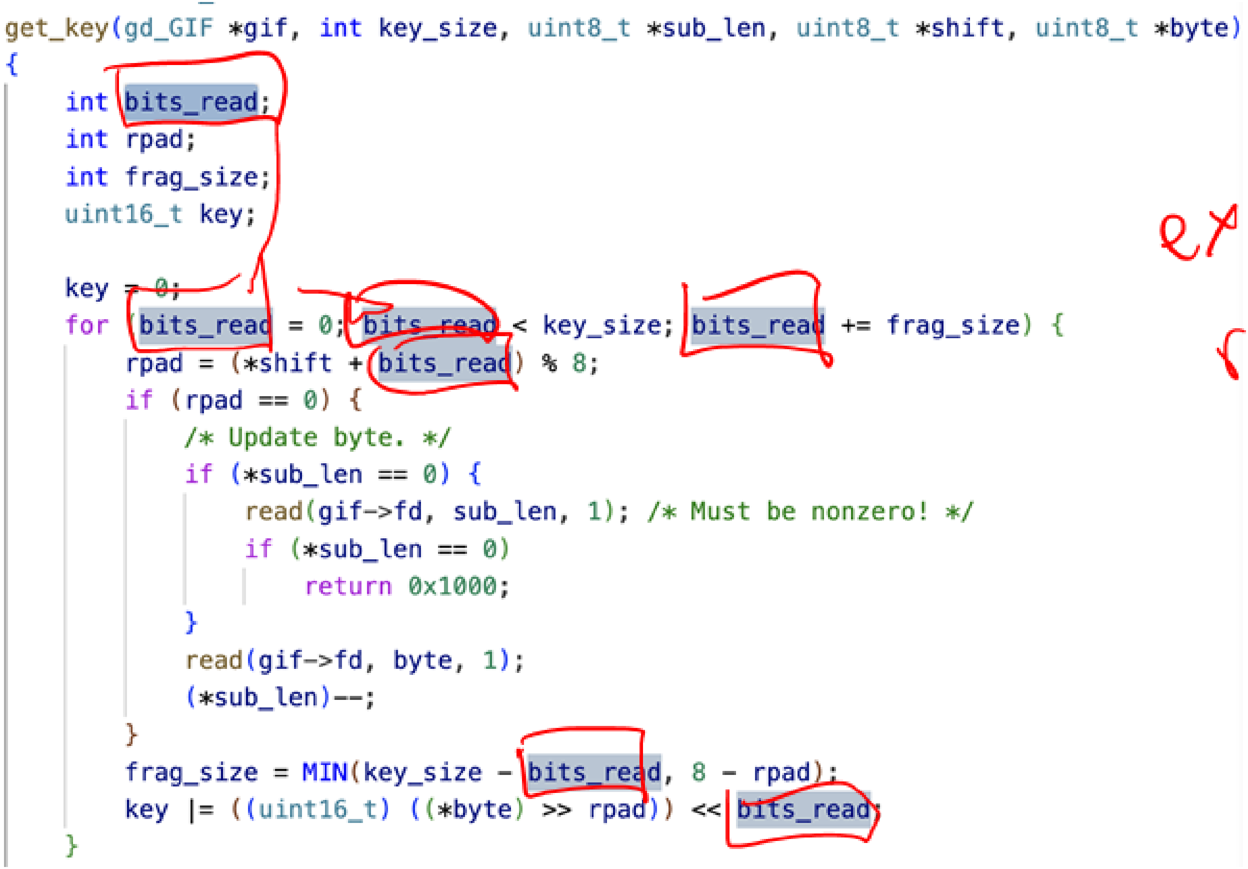
Caching - an algorithm for utilizing fast small memory and slow big memory.

Locality - the idea that computers often use nearby and similar information sequentially.

Spatial locality - nearby

Temporal locality - recently

Temporal Locality Example from MP 1

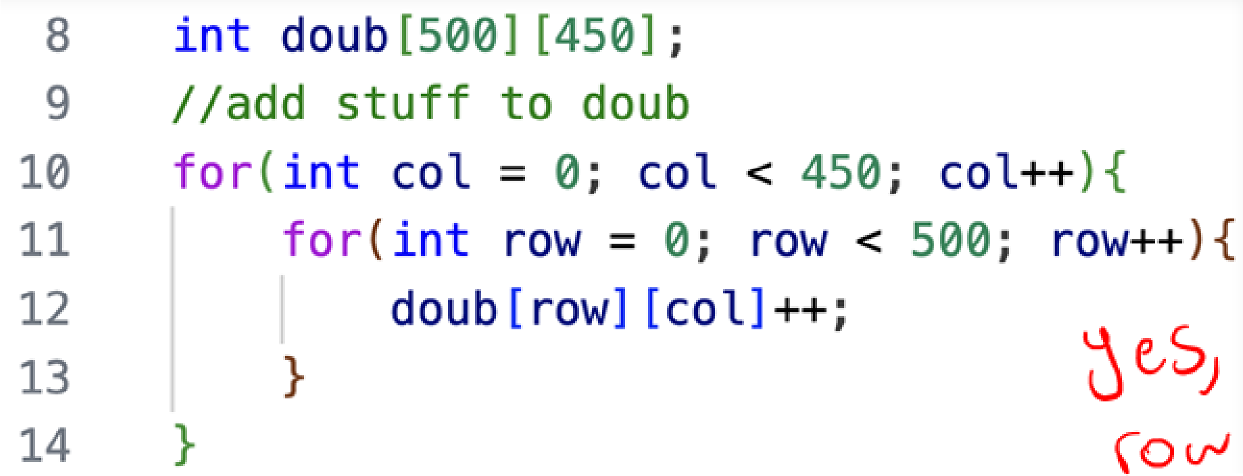


```
get_key(gd_GIF *gif, int key_size, uint8_t *sub_len, uint8_t *shift, uint8_t *byte)
{
    int bits_read;
    int rpad;
    int frag_size;
    uint16_t key;

    key = 0;
    for (bits_read = 0; bits_read < key_size; bits_read += frag_size) {
        rpad = (*shift + bits_read) % 8;
        if (rpad == 0) {
            /* Update byte. */
            if (*sub_len == 0) {
                read(gif->fd, sub_len, 1); /* Must be nonzero! */
                if (*sub_len == 0)
                    return 0x1000;
            }
            read(gif->fd, byte, 1);
            (*sub_len)--;
        }
        frag_size = MIN(key_size - bits_read, 8 - rpad);
        key |= ((uint16_t) ((*byte) >> rpad)) << bits_read;
    }
```

example of recency

How can you change this code for better locality?



```
int doub[500][450];
//add stuff to doub
for(int col = 0; col < 450; col++){
    for(int row = 0; row < 500; row++){
        doub[row][col]++;
    }
}
```

yes, swap col and row loops

clicker.cs.illinois.edu

Q9

~Code~
340