What is your experience with C?

clicker.cs.illinois.edu

Q1

~Code~
340

# CS 340

C without the ++

(C++ does not count!)

also mpl does not count!

# What percentage of the class do you think answered “Heard of C but haven’t used it”

clicker.cs.illinois.edu

Q2

~Code~
340

68%

[illegible] 15

# Updates

1. MP 0 - Setup due Tuesday.
    a. You may need 2-3 days to get it working
2. MP 1 - debugger due Tuesday
3. HW 2 Due NEXT Thursday 1:59pm
    a. Overview and C coding

mp2

mp3 –

# Learning C from C++

**Today's LGs:**

- Be able to compare C++ to C
- Be able to read and understand C code
- Be able to write C code from scratch

1. Why C
2. No Classes, new or delete, strings, cout
3. Demo

# Why C?

# How is C different from C++?

1. No templates
2. No classes
3. No overloading
4. No new or delete
5. No pass-by-reference
6. No standard C++ library

no strings
vectors
maps   cin cout

# How confused/nervous are you to code in C without the ++ features?

not at all
very

yes structs

# ~~No classes~~

~~no no~~ no member functions

no overloading

no templates

no pass-by-reference

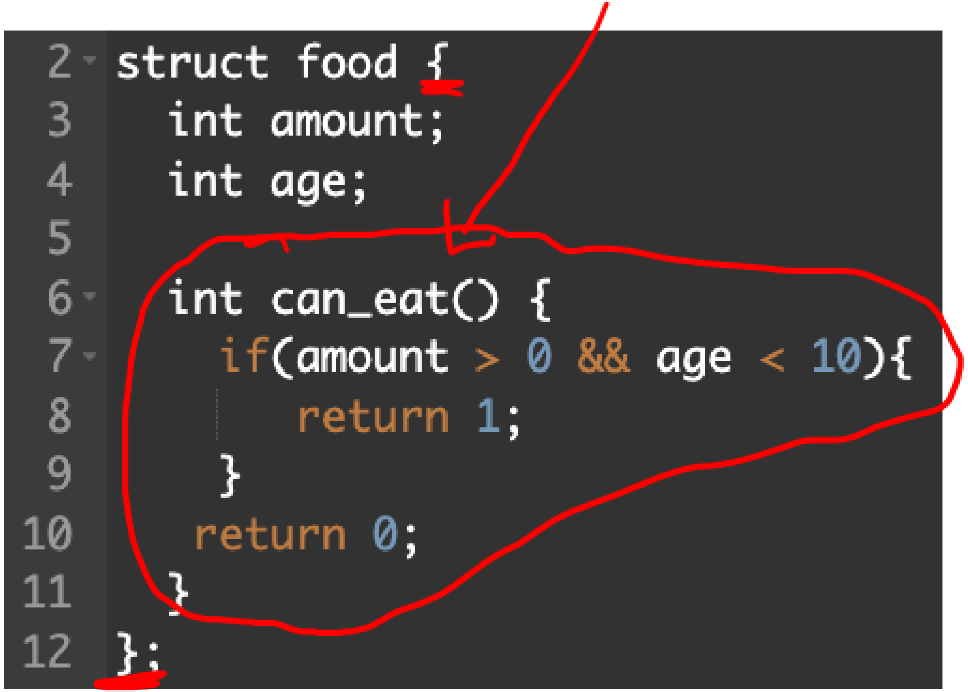
# Is this valid C code?

clicker.cs.illinois.edu

Q3

~Code~
340

what is this?



```c
struct food {
  int amount;
  int age;

  int can_eat() {
    if(amount > 0 && age < 10){
      return 1;
    }
   return 0;
  }
};
```

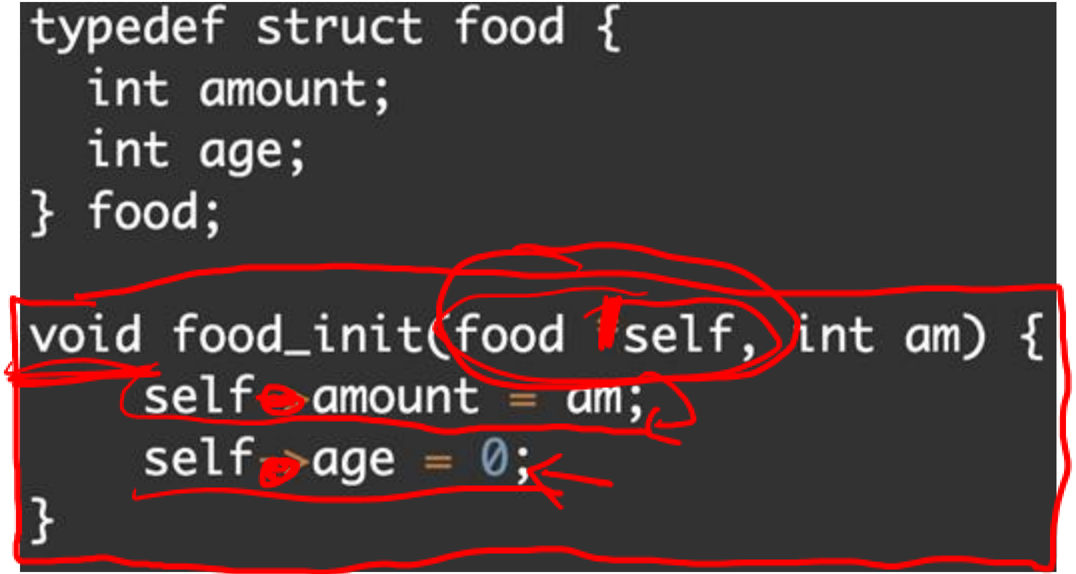
NO

# Structs strategy/style

1. Create a struct with ONLY member variables
2. Write functions that work with a pointer to an instance of the struct

```
typedef struct food{
    int amount;
    int age;
} food;
```

```
int can_eat(food *self){
    if(self->age <10){
        return 1;
    }
    return 0;
}
```



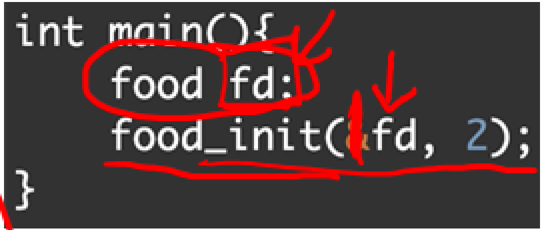
```c
typedef struct food {
   int amount;
   int age;
} food;

void food_init(food *self, int am) {
     self->amount = am;
     self->age = 0;
}
```

clicker.cs.illinois.edu

Q4

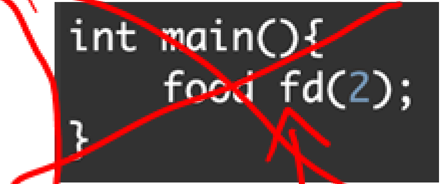
~Code~
340

A



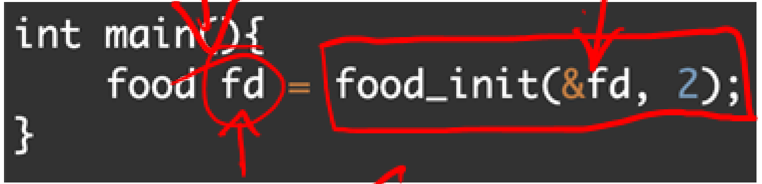
```c
int main(){
     food fd;
     food_init(&fd, 2);
}
```

B



```c
int main(){
     food fd(2);
}
```

C



```c
int main(){
     food fd = food_init(&fd, 2);
}
```

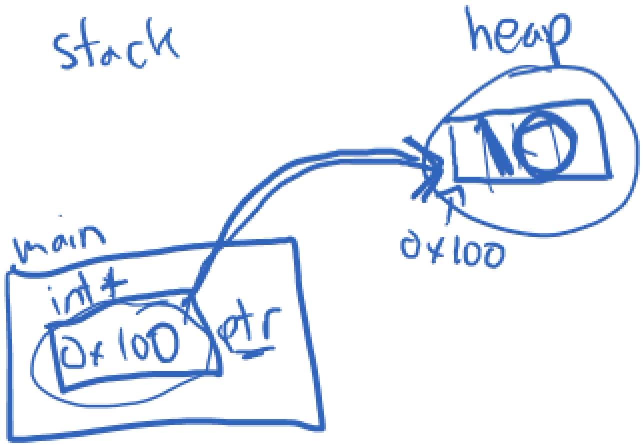
yes malloc free

# ~~No new or delete~~

```
//allocates size bytes on the heap and returns a
//pointer to that memory location on the heap.
void *malloc(size_t size);
```

8 bits

```
//frees the memory at ptr from the heap
void free(void *ptr);
```



4

```
int main(){
int* ptr = malloc(sizeof(int));
*ptr = 10;
free(ptr);
~~printf~~
```

NULL

# What are some common memory management errors?

double free

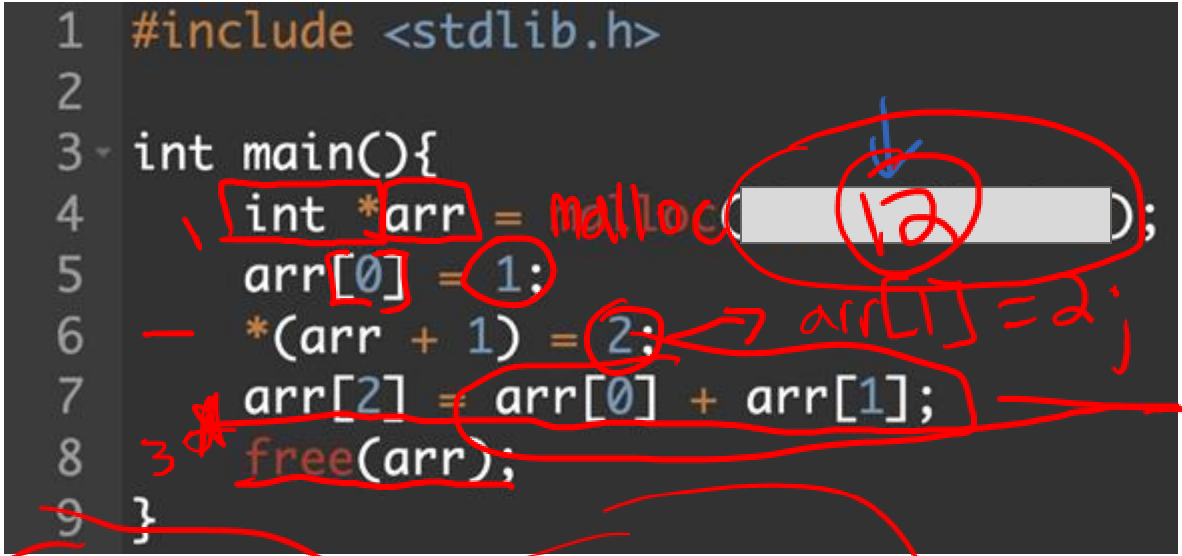
memory leak — orphaned memory

bad memory

stack overflow

bad free

# What should go in the box?



```c
#include <stdlib.h>

int main(){
    int *arr = malloc(          );
    arr[0] = 1;
    *(arr + 1) = 2;
    arr[2] = arr[0] + arr[1];
    free(arr);
}
```

12

arr[1] = 2;

3

*arr → 1

*(arr+1) → 2

arr[1] → 2

sizeof(int) * 3

calloc(3, sizeof(int)); → init to 0

- Type?
int = 4 bytes

4bytes

|   |   |   |
|---|---|---|
| 1 | 2 | 3 |

clicker.cs.illinois.edu

Q5

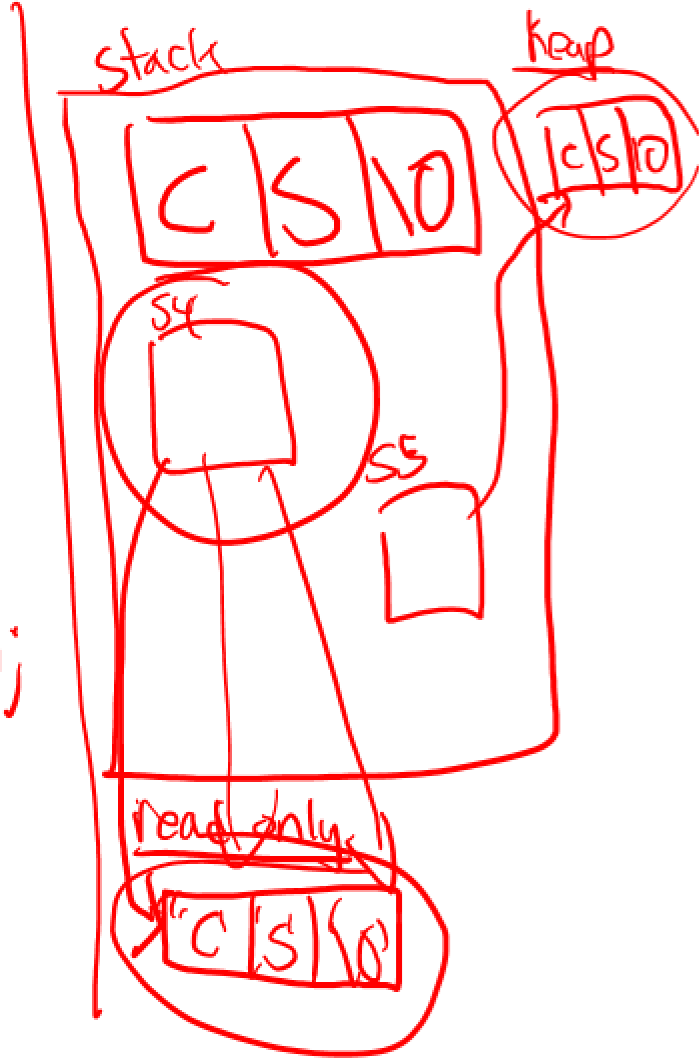
~Code~
340

Yes c-strings

# ~~No strings~~

# C-Strings

```
int main(){
char s3[3] = {'C', 'S', '\0'};
char s2[3] = "CS";
char* s4 = "CS"
char* s5 = malloc(sizeof(char)*3);
~~strcpy~~(s5, "CS"))
strcpy
```



# #include <string.h>

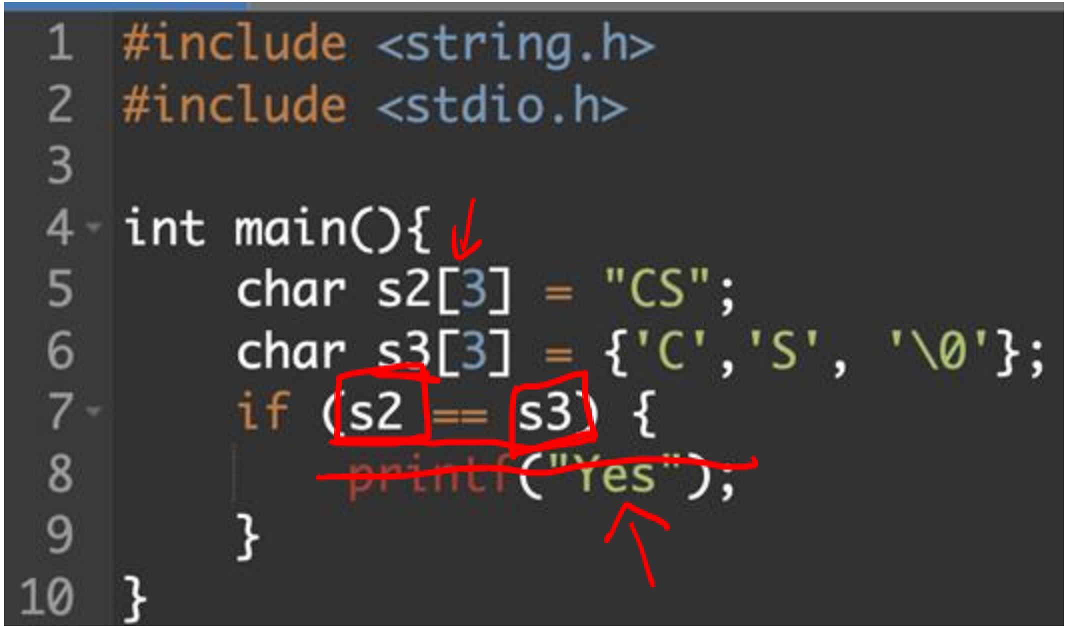
- strlen()
- strcpy()
- strcmp()
- ...more

# What prints? Why?

clicker.cs.illinois.edu

06

~Code~
340

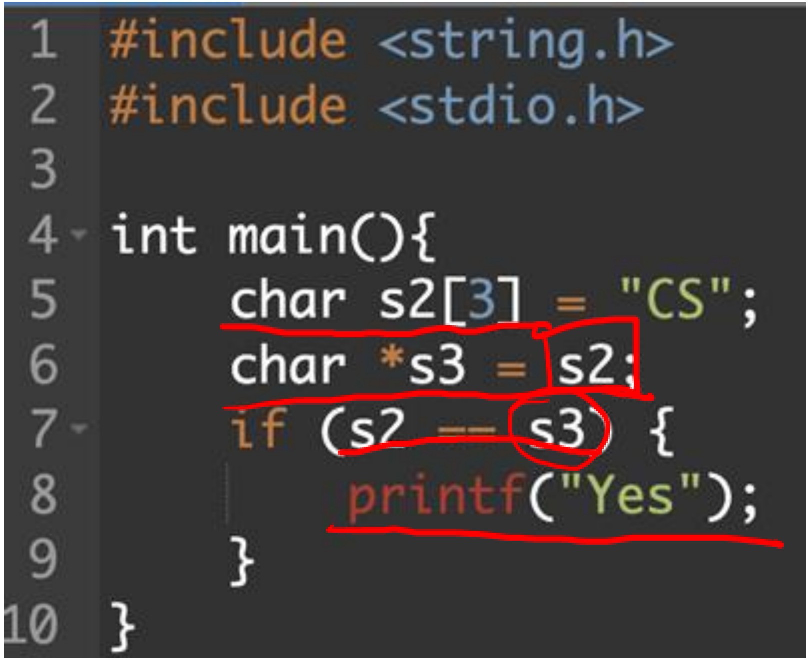


```
#include <string.h>
#include <stdio.h>

int main(){
      char s2[3] = "CS";
      char s3[3] = {'C','S', '\0'};
      if (s2 == s3) {
        ~~printf("Yes");~~
      }
}
```

strcmp(s2,s3) nothing

|  |  |  |  |
|---|---|---|---|
| s2<br>0x100 | 'C' | 'S' | '\0' |
| s3<br>0x500 | 'C' | 'S' | '\0' |

# What prints? Why?



```c
#include <string.h>
#include <stdio.h>

int main(){
    char s2[3] = "CS";
    char *s3 = s2;
    if (s2 == s3) {
        printf("Yes");
    }
}
```

s3 → 0x100

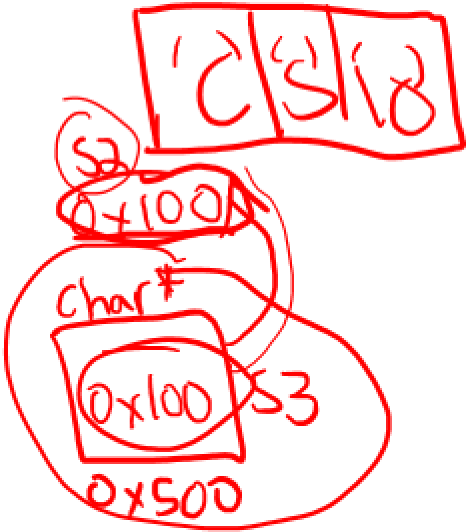
*s3 → 'C'

&s3 → 0x500

clicker.cs.illinois.edu

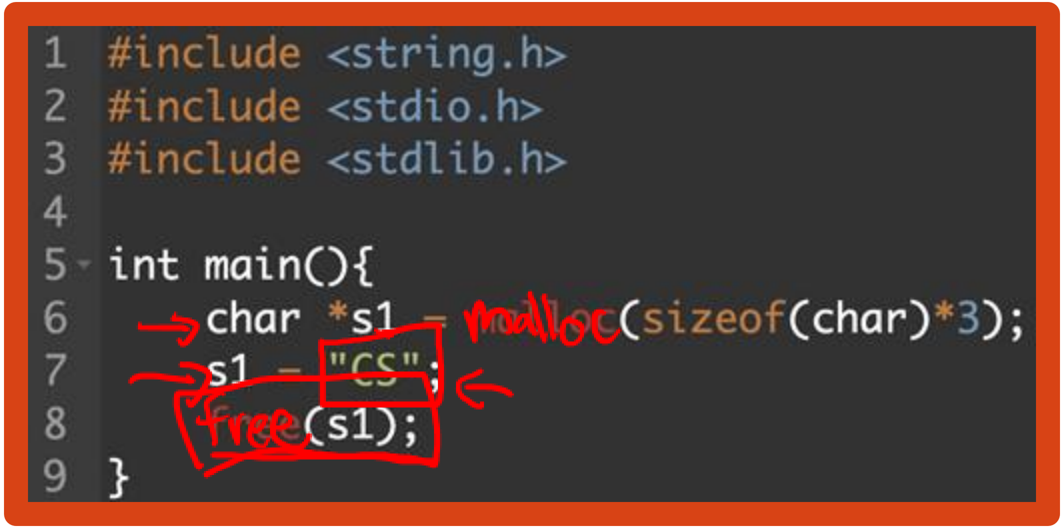
Q7

~Code~
340

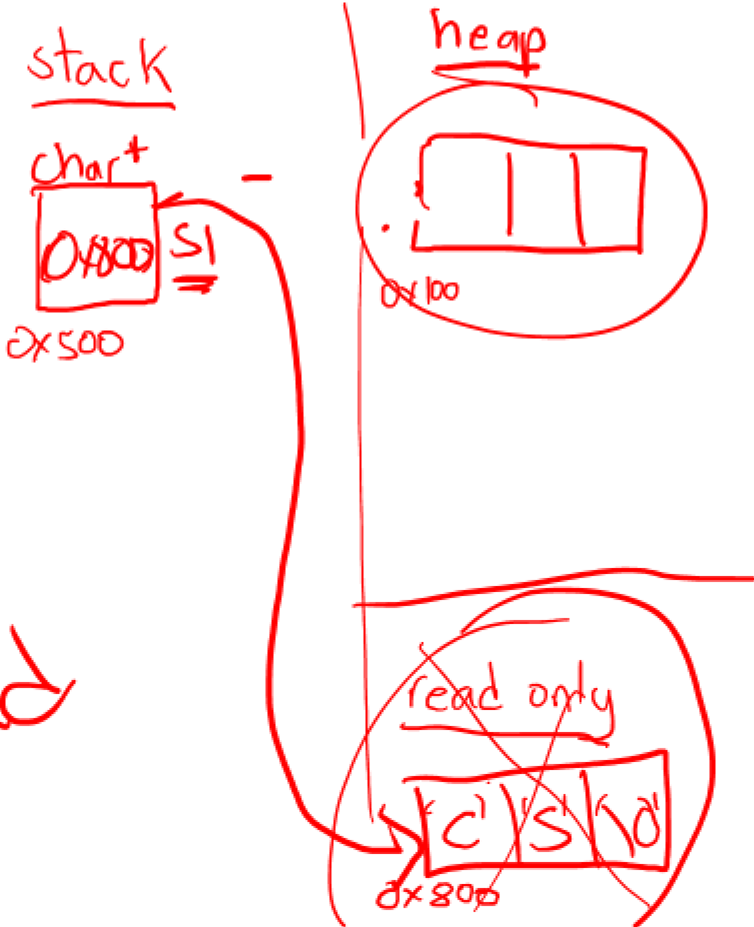


yes

# What is the issue with the top code snippet?

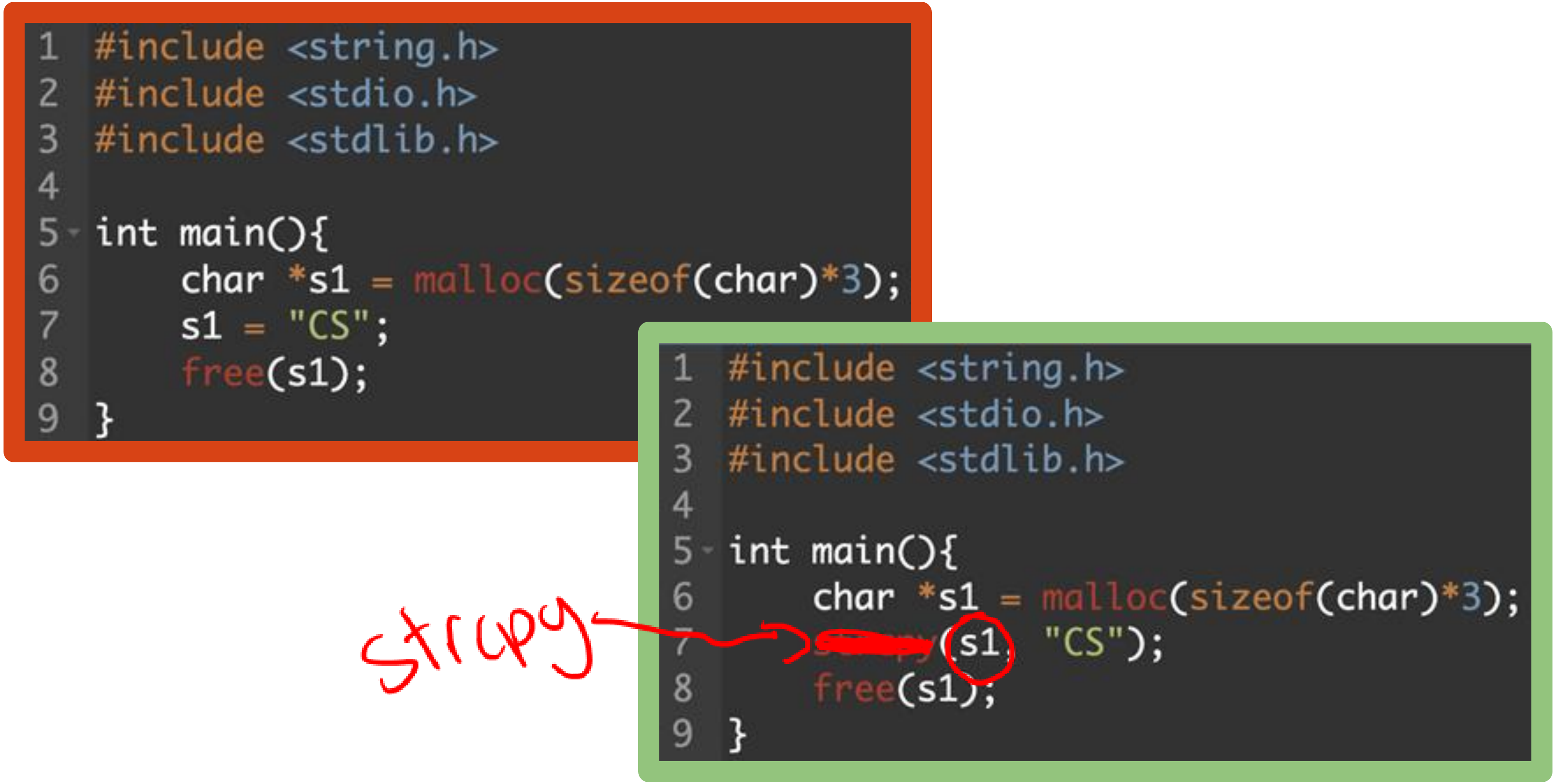


```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(){
    char *s1 = malloc(sizeof(char)*3);
    s1 = "CS";
    free(s1);
}
```



Bad

# What is the issue with the top code snippet?



```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(){
    char *s1 = malloc(sizeof(char)*3);
    s1 = "CS";
    free(s1);
}
```

```c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main(){
    char *s1 = malloc(sizeof(char)*3);
    strcpy(s1, "CS");
    free(s1);
}
```

strcpy

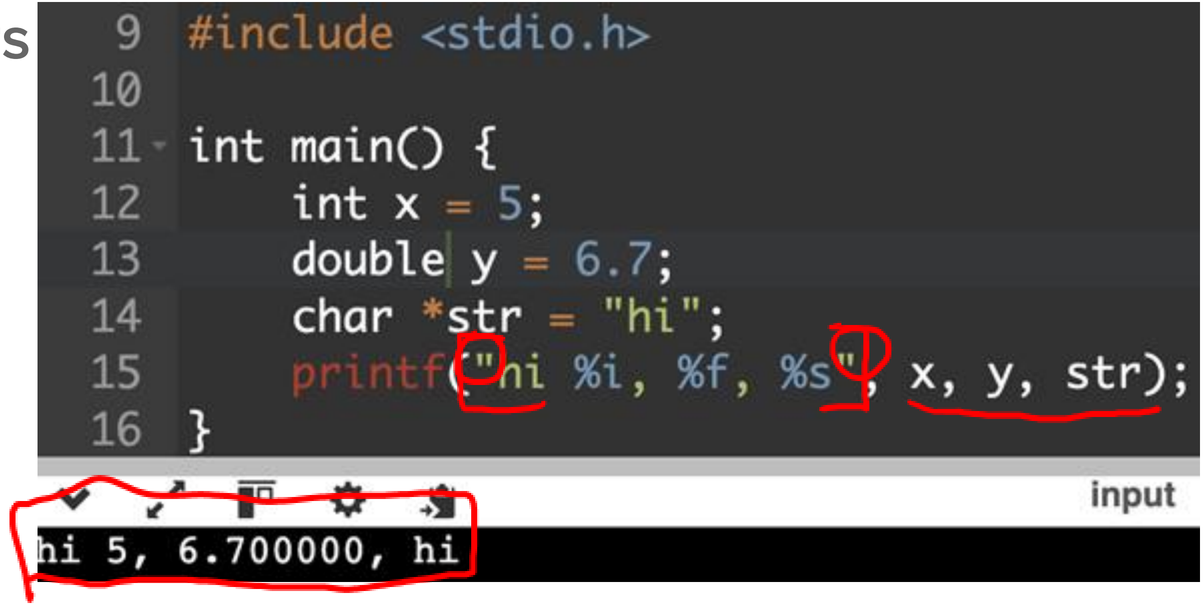
# Need a refresher on pointers/c-strings/arrays?

Fill out this form to be added to the free coursera course!

# No cout

# int printf(“format string”, args);

1. %d or %i: for integers
2. %f: for floating-point numbers (floats and doubles)
3. %c: for characters
4. %s: for C strings
5. ...more

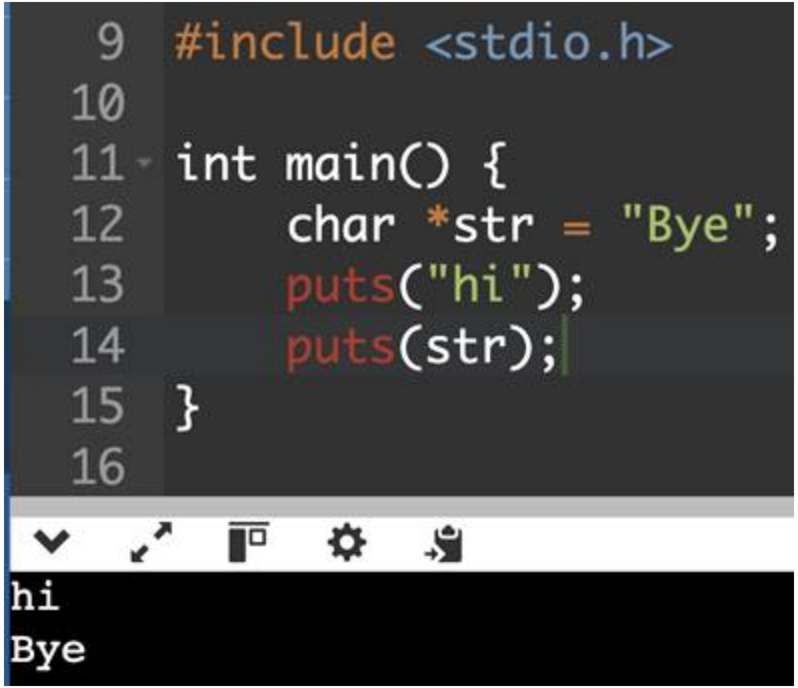


```c
#include <stdio.h>

int main() {
    int x = 5;
    double y = 6.7;
    char *str = "hi";
    printf("hi %i, %f, %s", x, y, str);
}
```

```
hi 5, 6.700000, hi
```

# int puts(const char *str);

Just C-Strings!



```c
#include <stdio.h>

int main() {
      char *str = "Bye";
      puts("hi");
      puts(str);
}

```

```
hi
Bye
```

# No vectors, maps, linked-lists, sets… ect

yes

# ~~No~~ Demo

# Your Turn