CSE 29
Lecture 1 Summary

January 6, 2026

Review Check Question(s)

Answer 1: Write down the name of someone you met in class

Answer 2: Write the binary representation of ‘A’

Answer 3: Questions you have for Joe

Course Logistics

Please visit the course website to view the course syllabus

ucsd-cse29.github.io/wi26

AI Policy

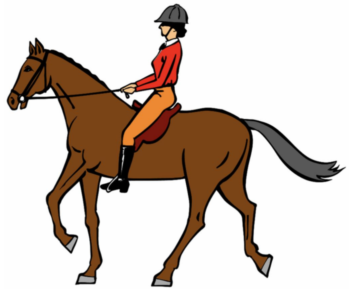
You should not use AI to generate code or prose for work in this class.

WHY?

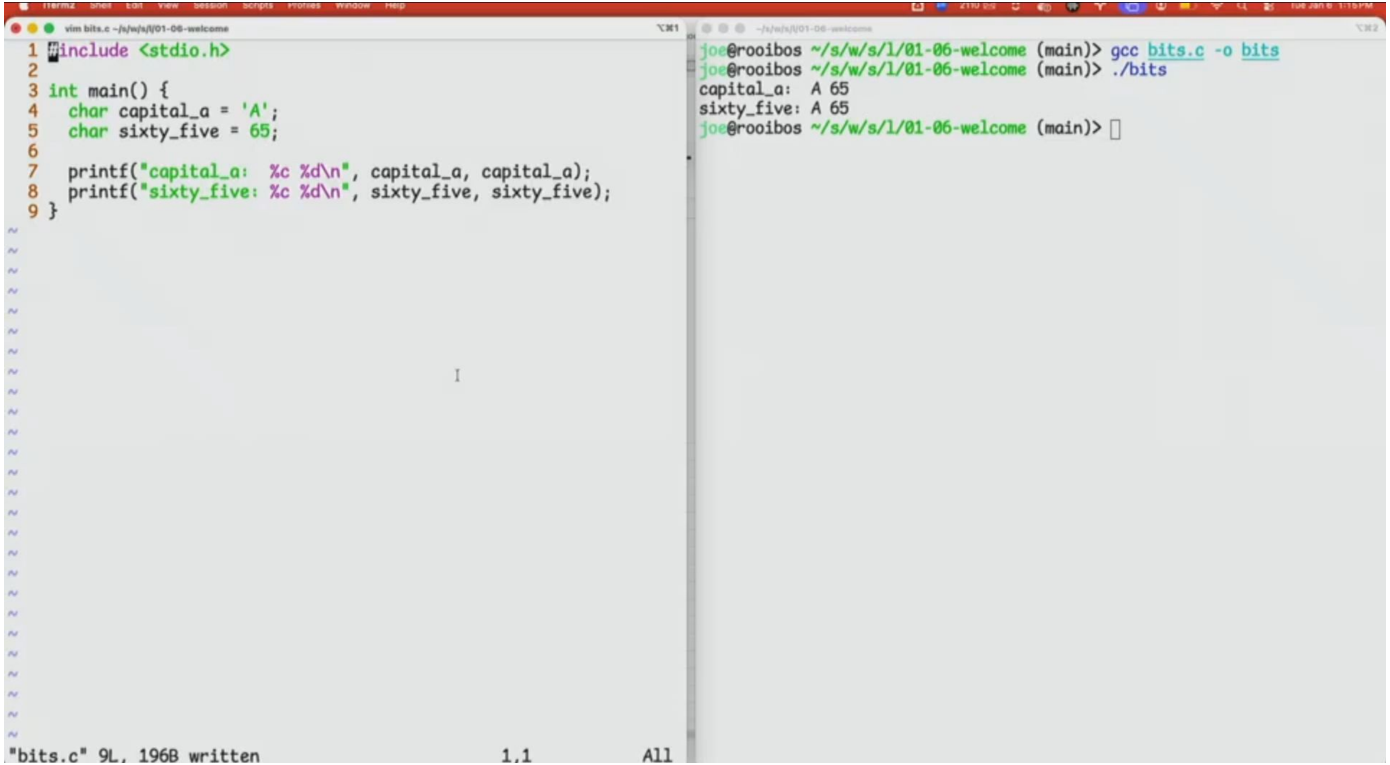
Programming on your own is like how you run for health and fitness, not for transportation.

A scooter (or horse) can you anywhere faster, but misses the point. Running for fitness is about **internal changes**, not the destination. Similarly, programming internally changes you, through **learning**.

…plus, exams require that you program on your own.



What questions/observations do you have when you look at this?



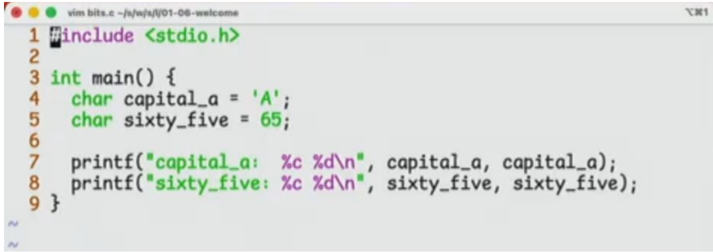
```
#include <stdio.h>

int main() {
  char capital_a = 'A';
  char sixty_five = 65;

  printf("capital_a:  %c %d\n", capital_a, capital_a);
  printf("sixty_five: %c %d\n", sixty_five, sixty_five);
}
```

```
"bits.c" 9L, 196B written                    1,1           All
```

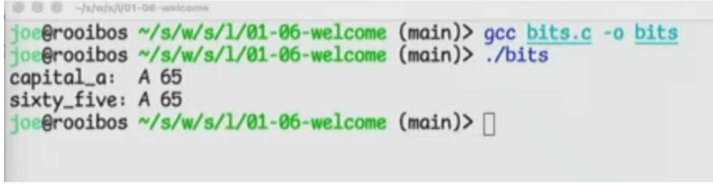
```
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> gcc bits.c -o bits
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> ./bits
capital_a:  A 65
sixty_five: A 65
joe@rooibos ~/s/w/s/l/01-06-welcome (main)>
```

Questions/observations from students

- In line 5, we defined a character using an integer
- The `main` function has a `int` return type but no value being returned
  - In the C programming language, `main` is a special function and is treated as it returns `0`, despite there being no return statement
- Can the return type of `main` be any type?
  - No, it must be `int`
- `capital_a` stores `’A’` but can be printed as both `’A’` and `65` in line 7
  - Same with `sixty_five`, printed in line 8
- What is the percent?
  - `%c` means **print formatted as (ASCII) character**
  - `%d` means **print formatted as a decimal number**



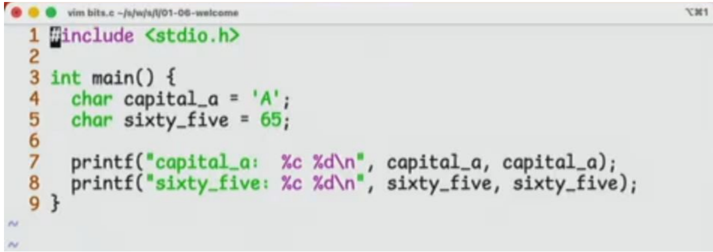
```
1 #include <stdio.h>
2
3 int main() {
4   char capital_a = 'A';
5   char sixty_five = 65;
6
7   printf("capital_a:  %c %d\n", capital_a, capital_a);
8   printf("sixty_five: %c %d\n", sixty_five, sixty_five);
9 }
~
~
```



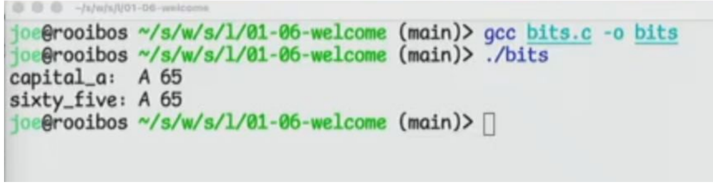
```
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> gcc bits.c -o bits
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> ./bits
capital_a:  A 65
sixty_five: A 65
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> 
```

Questions/observations from students cont.

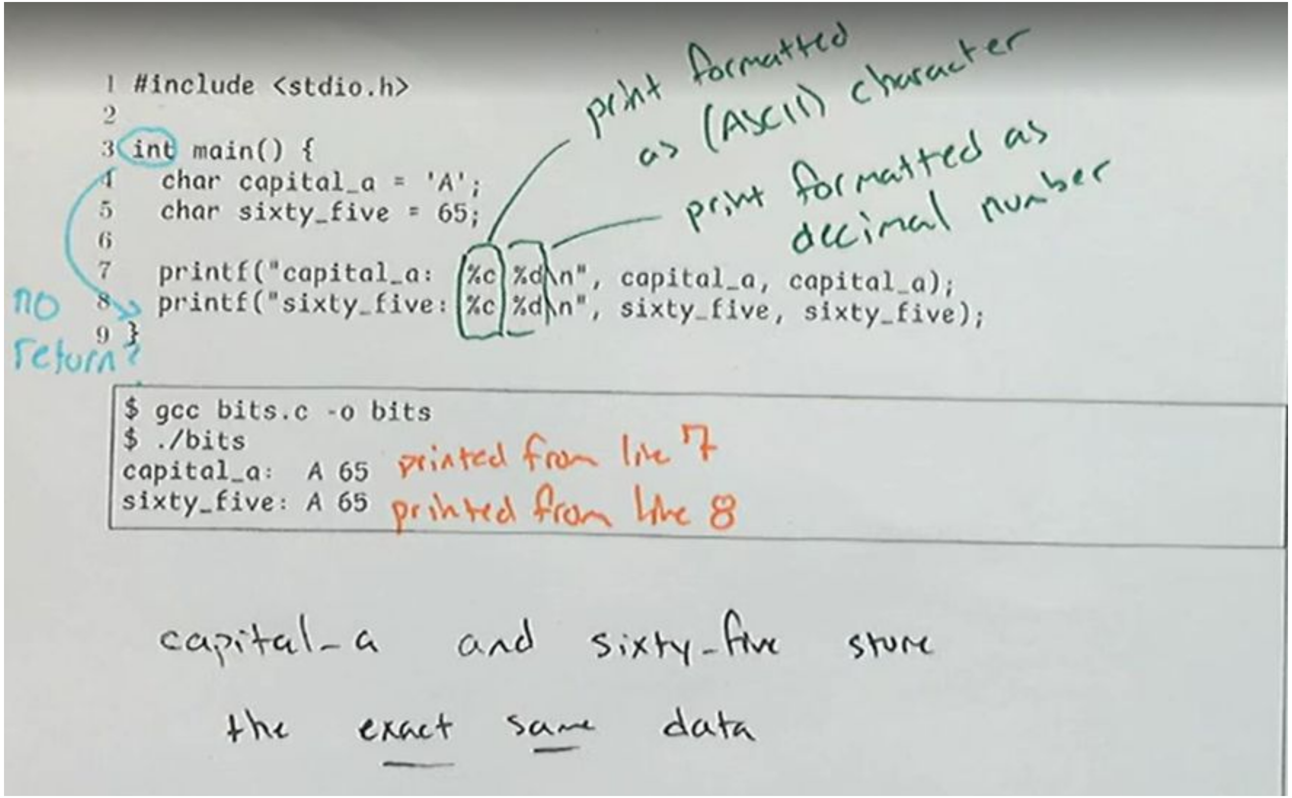
- How would we get a type error in C?
  - On line 5, we initialized a char variable with an integer, which did not return a type error
  - If we tried to do string concatenation like: `“hello” + “world”` that would be a type error because strings cannot be added in C
- `#include <stdio.h>` is what defines `printf` and other input/output operations
- `‘A’ == 65` would evaluate to 1 (true)



```
1 #include <stdio.h>
2
3 int main() {
4   char capital_a = 'A';
5   char sixty_five = 65;
6
7   printf("capital_a:  %c %d\n", capital_a, capital_a);
8   printf("sixty_five: %c %d\n", sixty_five, sixty_five);
9 }
```



```
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> gcc bits.c -o bits
joe@rooibos ~/s/w/s/l/01-06-welcome (main)> ./bits
capital_a:  A 65
sixty_five: A 65
joe@rooibos ~/s/w/s/l/01-06-welcome (main)>
```



```
1 #include <stdio.h>
2
3 int main() {
4   char capital_a = 'A';
5   char sixty_five = 65;
6
7   printf("capital_a:  %c %d\n", capital_a, capital_a);
8   printf("sixty_five: %c %d\n", sixty_five, sixty_five);
9 }
```

print formatted as (ASCII) character

print formatted as decimal number

no return?

```
$ gcc bits.c -o bits
$ ./bits
capital_a:  A 65
sixty_five: A 65
```

printed from line 7

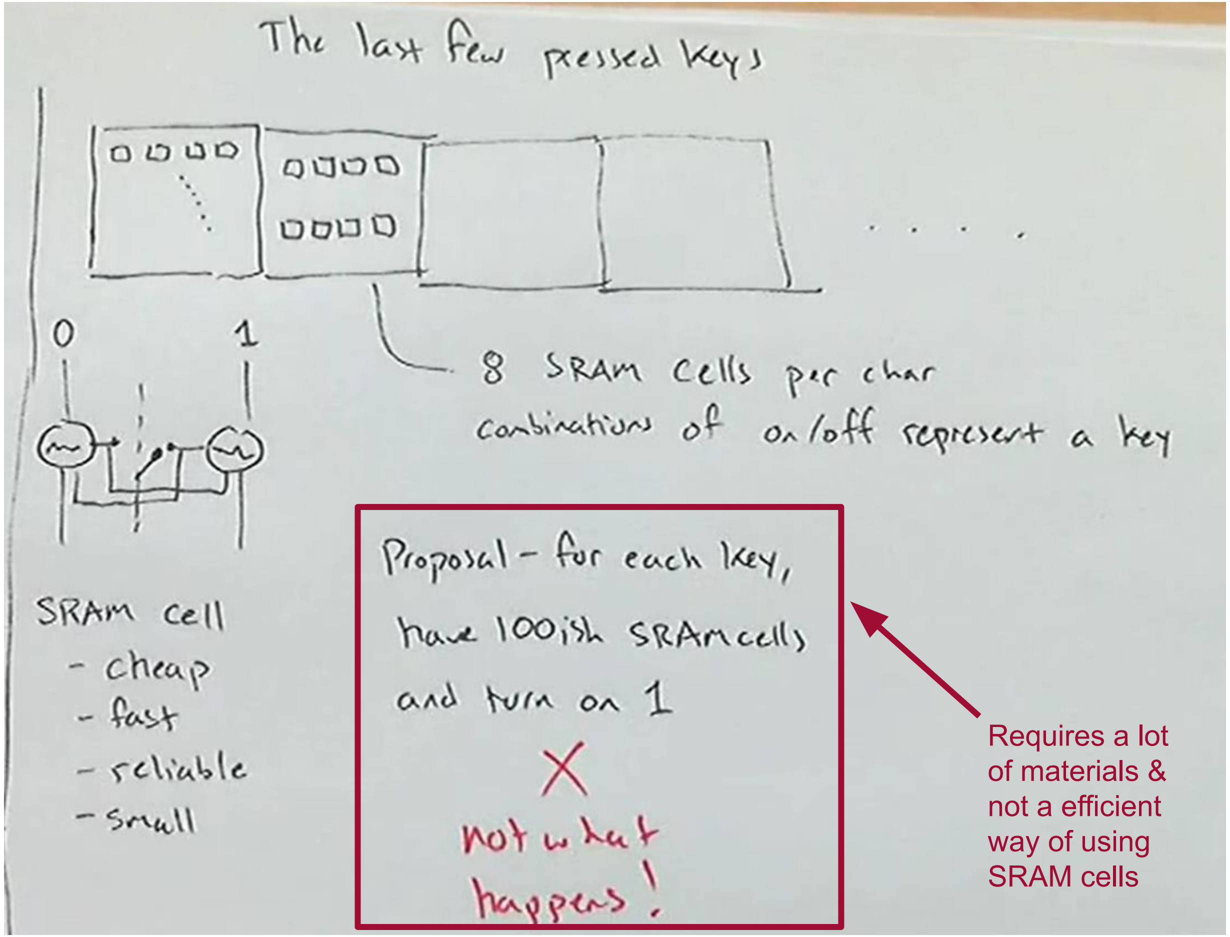
printed from line 8

capital_a and sixty_five store the exact same data

What happens when you press a key on a keyboard?

When I push the ‘A’ key on a keyboard…

- A circuit is closed and electricity flows somewhere in to the computer
- The operating system (OS) share physical storage and circuitry to keep track of the last few keys that were pressed (could be last 10, last 1000, it’s up to the OS)
    - Programs (Chrome, a PC game, etc.) can access this
- SRAM cells (made of metal or silicon) have a toggle that can stick to one of two respective sides (depending on where electricity is flowing to)
    - One side is a 0 signal and the other is a 1 signal
    - This stores **one bit**
- 8 SRAM cells make up one key being stored



The last few pressed keys

8 SRAM cells per char
combinations of on/off represent a key

SRAM cell
- cheap
- fast
- reliable
- small

Proposal - for each key,
have 100ish SRAM cells
and turn on 1

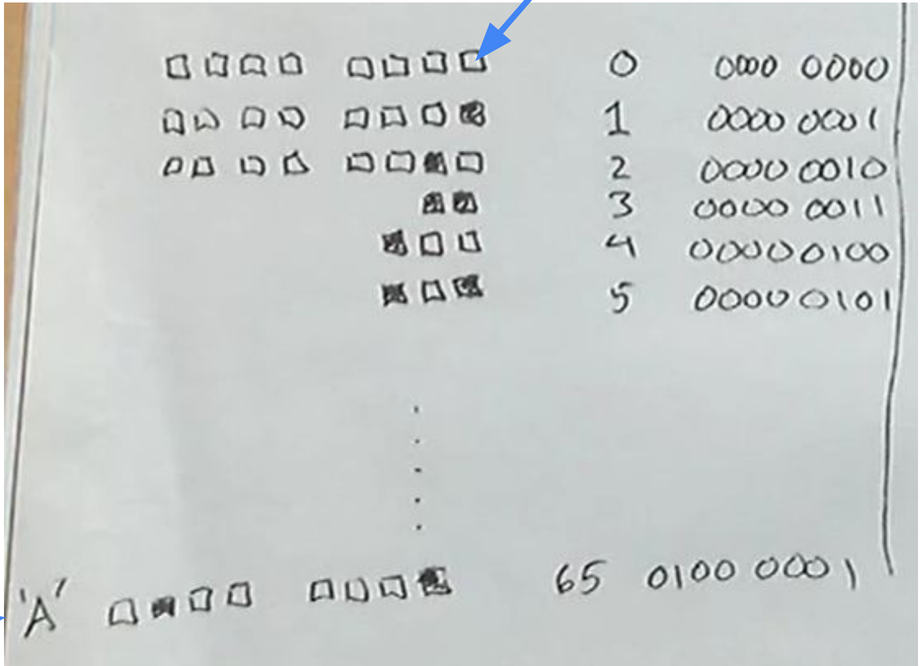
not what happens!

Requires a lot
of materials &
not a efficient
way of using
SRAM cells

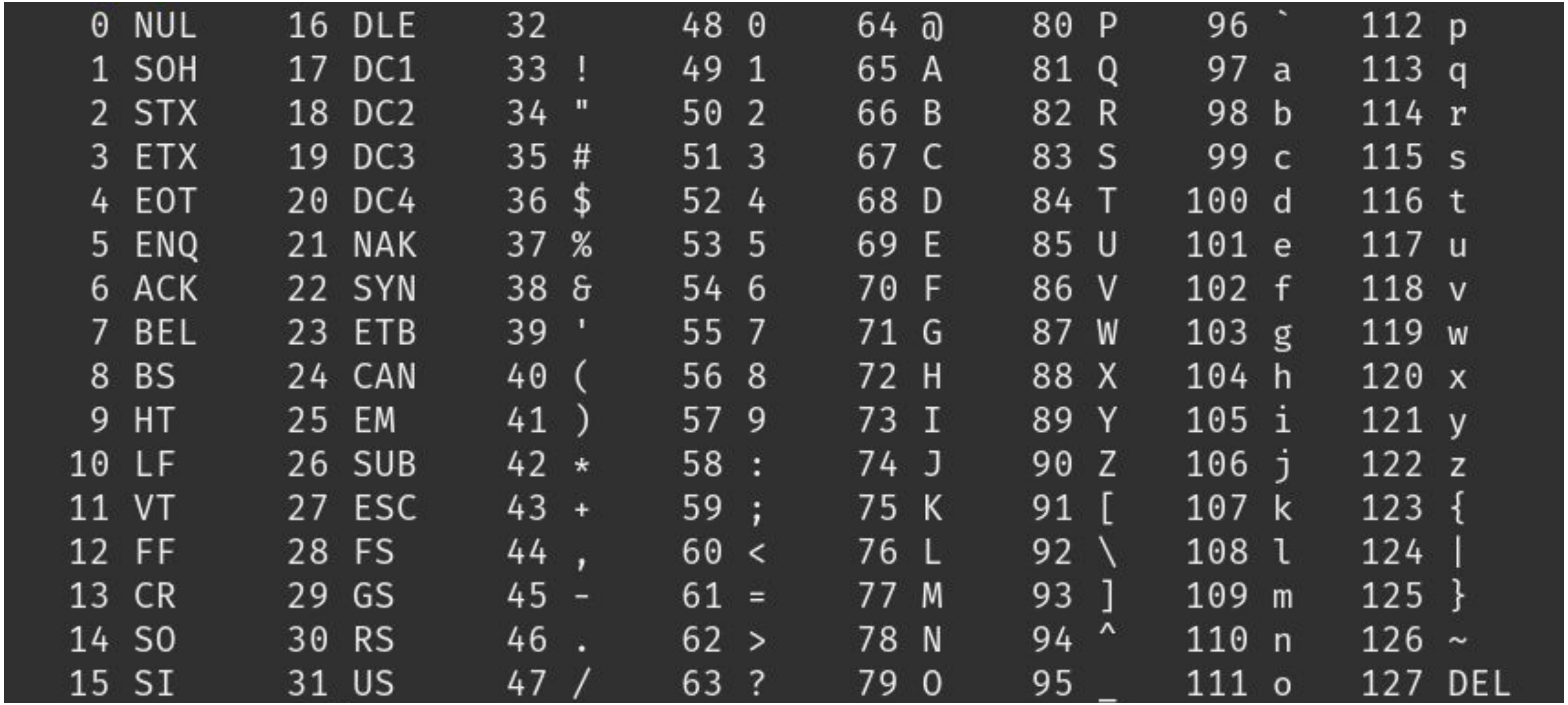
Values Represented in Hardware

- When a keyboard key is pressed, the computer stores a sequence of 0’s and 1’s using electricity
- We can use 0’s and 1’s to represent numbers using a base-2 (binary) number system
  - `0000 0000` in base-2 is `0` in base 10
  - `0100 0001` in base 2 is `65` in base 10
- `capital_a` and `sixty_five` are stored the same way in a computer

One SRAM cell (off)



ASCII Table



```
  0 NUL    16 DLE    32        48 0    64 @    80 P     96 `   112 p
  1 SOH    17 DC1    33 !      49 1    65 A    81 Q     97 a   113 q
  2 STX    18 DC2    34 "      50 2    66 B    82 R     98 b   114 r
  3 ETX    19 DC3    35 #      51 3    67 C    83 S     99 c   115 s
  4 EOT    20 DC4    36 $      52 4    68 D    84 T    100 d   116 t
  5 ENQ    21 NAK    37 %      53 5    69 E    85 U    101 e   117 u
  6 ACK    22 SYN    38 &      54 6    70 F    86 V    102 f   118 v
  7 BEL    23 ETB    39 '      55 7    71 G    87 W    103 g   119 w
  8 BS     24 CAN    40 (      56 8    72 H    88 X    104 h   120 x
  9 HT     25 EM     41 )      57 9    73 I    89 Y    105 i   121 y
 10 LF     26 SUB    42 *      58 :    74 J    90 Z    106 j   122 z
 11 VT     27 ESC    43 +      59 ;    75 K    91 [    107 k   123 {
 12 FF     28 FS     44 ,      60 <    76 L    92 \    108 l   124 |
 13 CR     29 GS     45 -      61 =    77 M    93 ]    109 m   125 }
 14 SO     30 RS     46 .      62 >    78 N    94 ^    110 n   126 ~
 15 SI     31 US     47 /      63 ?    79 O    95 _    111 o   127 DEL
```

Common exit slip questions

Thank you for your questions and feedback! 🙂

How to represent with bigger numbers/other characters?

- For bigger numbers, we will see C types that are made of multiple bytes
- For other characters, that will be the focus of upcoming lectures

Do binary representations change with each language?
What is the left most bit used for in ASCII?

- Will be answered when we talk about Unicode/UTF (coming soon)

Is there a lecture recording?

- We don't plan to publish a lecture recording

Java vs C vs C++ differences, C syntax, Print statements in C

- We have a textbook! See https://diveintosystems.org/book/Appendix1/index.html in the textbook for Java comparisons, for example

Why 8 bits? / Why only 8 cells and not another number?

- Mostly just a good engineering tradeoff. 8 bits is enough to represent "a lot" without wasting "a lot" of space. Also, for a lot of reasons computer hardware works better with powers of 2, which is why 8 and not 10. Larger data takes up multiple bytes.

Do we need to know SRAM for exam?

- I generally don't answer questions about the exam content ahead of time.

Why is there no return statement in main?

- Main is special and implicitly returns 0 if there is no provided return

What does -o in compile mean?

- "Output" – the next argument given to the command is the output file to store the compiled program in