CMSC201
Computer Science I for Majors

Lecture 22 – Data Representation

Last Class We Covered

- Sorting
  - Bubble
  - Selection
  - Quick
- Searching
  - Linear
  - Binary

Any Questions from Last Time?

Today's Objectives

- To understand how data is represented and stored in memory
  - Binary numbers
  - Hexadecimal numbers
  - Converting
    - Binary to Decimal
    - Decimal to Binary
  - ASCII

Binary Numbers

Binary Numbers

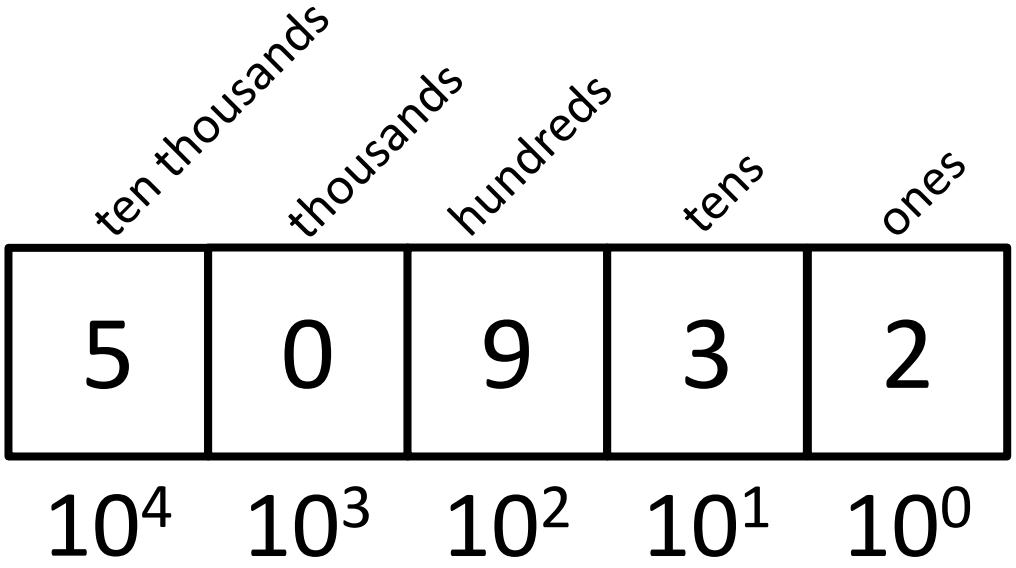
- Computers store all information (code, text, images, sound,) as a binary representation
  - "Binary" means only two parts: 0 and 1
- Specific formats for each file help the computer know what type of item/object it is
- But why use binary?

Decimal vs Binary

- Why do we use decimal numbers?
  - Ones, tens, hundreds, thousands, etc.
- But computers don't have fingers…
  - What do they have instead?
- They only have two states: "on" and "off"

Decimal Example

- How do we represent a number like 50,932?



$2 \times 10^0 = 2$
$3 \times 10^1 = 30$
$9 \times 10^2 = 900$
$0 \times 10^3 = 0000$
$5 \times 10^4 = 50000$

Total: 50932

Decimal uses 10 digits, so…

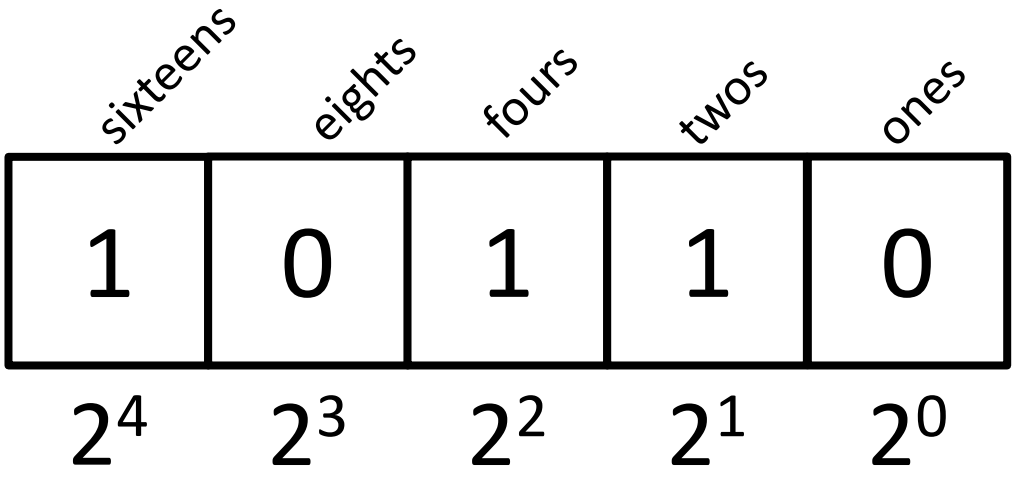
Another Decimal Example

6	7	4	9	3
10^4	10^3	10^2	10^1	10^0
10000	1000	100	10	1
60000	7000	400	90	3

60000+7000+400+90+3 = 67493

Binary Example

- Let's do the same with 10110 in binary



$0 \times 2^0 = 0$
$1 \times 2^1 = 2$
$1 \times 2^2 = 4$
$0 \times 2^3 = 0$
$1 \times 2^4 = 16$

Total: 22

Binary uses 2 digits, so our base isn't 10, but…

Binary to Decimal Conversion

- Step 1: Draw Conversion Box
- Step 2: Enter Binary Number
- Step 3: Multiply
- Step 4: Add

1	0	0	0	1	1	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
128	0	0	0	8	4	0	1

128 + 0 + 0 + 0 + 8 + 4 + 0 + 1 = 141

Exercise: Converting From Binary

- What are the decimals equivalents of…

```
101
1111
100000
101010
0010 1010
1000 0000
```

Longer binary numbers are often broken into blocks of four digits for the sake of readability

Exercise: Converting From Binary

- What are the decimals equivalents of…

```
101          = 4+0+1                = 5
1111         = 8+4+2+1              = 15
100000       = 32+0+0+0+0+0         = 32
101010       = 32+0+8+0+2+0         = 42
0010 1010    = 32+0+8+0+2+0         = 42
1000 0000    = 128+...+0+0          = 128
```

Decimal to Binary Conversion

- Step 1: Draw Conversion Box
- Step 2: Compare decimal to highest binary value
- Step 3: If binary value is smaller, put a 1 there and subtract the value from the decimal number
- Step 4: Repeat until 0

Convert 163 to binary

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	0	1	0	0	0	1	1

163-128 = 35 35-32 = 3 3-2=1 1-1=0

Converting to Binary

- What are the binary equivalents of…

```
9

27

68

216


255
```

Converting to Binary

- What are the binary equivalents of…

```
9     = 1001 (or 8+1)
27    = 0001 1011 (or 16+8+2+1)
68    = 0100 0100 (or 64+4)
216   = 1101 1000
        (or 128+64+16+8)
255   = 1111 1111
        (or 128+64+32+16+8+4+2+1)
```

Binary Tips and Tricks

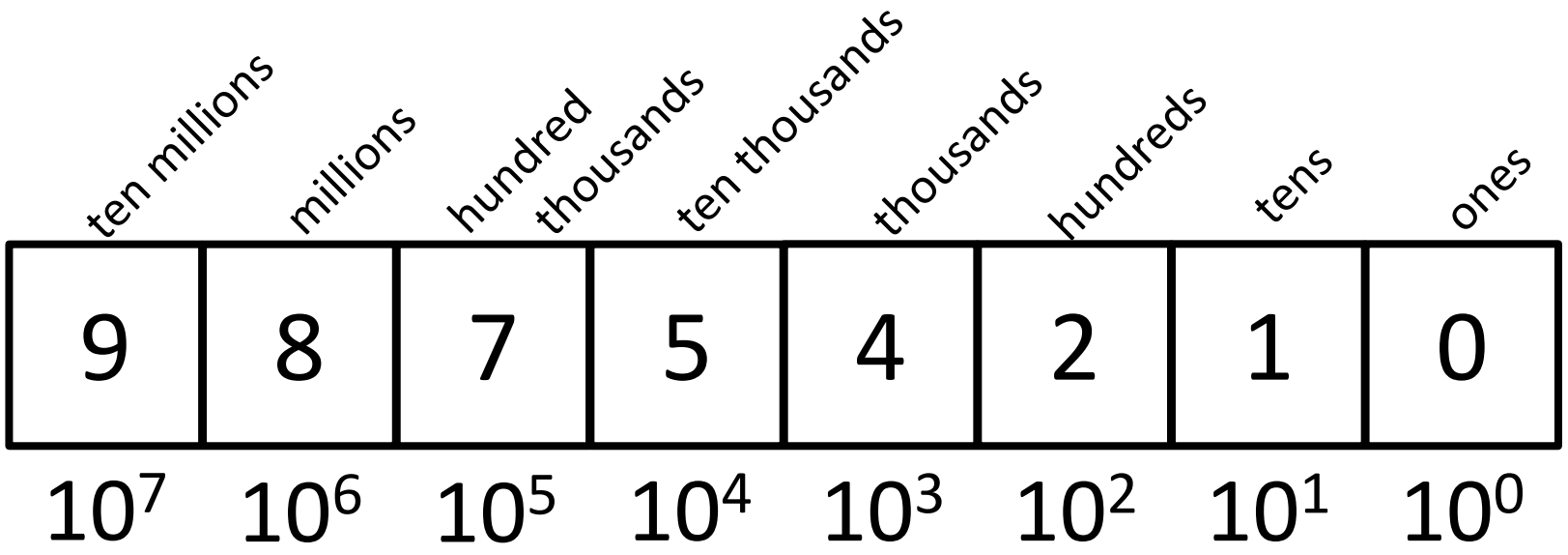
- Some "sanity checking" rules for conversions:

1. Binary can only be 1 or 0
   - If you get "2" of something, it's wrong
2. Odd numbers must have a 1 in the ones column
   - Why? (And what's the rule for even numbers?)
3. Each column's value is the sum of all of the previous columns (to the right) plus one
   - In decimal, what column comes after 999?

Hexadecimal Numbers

Decimal Representation

- Decimal uses 10 digits
  - Decimal, *deci* = 10
  - The digits used are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9

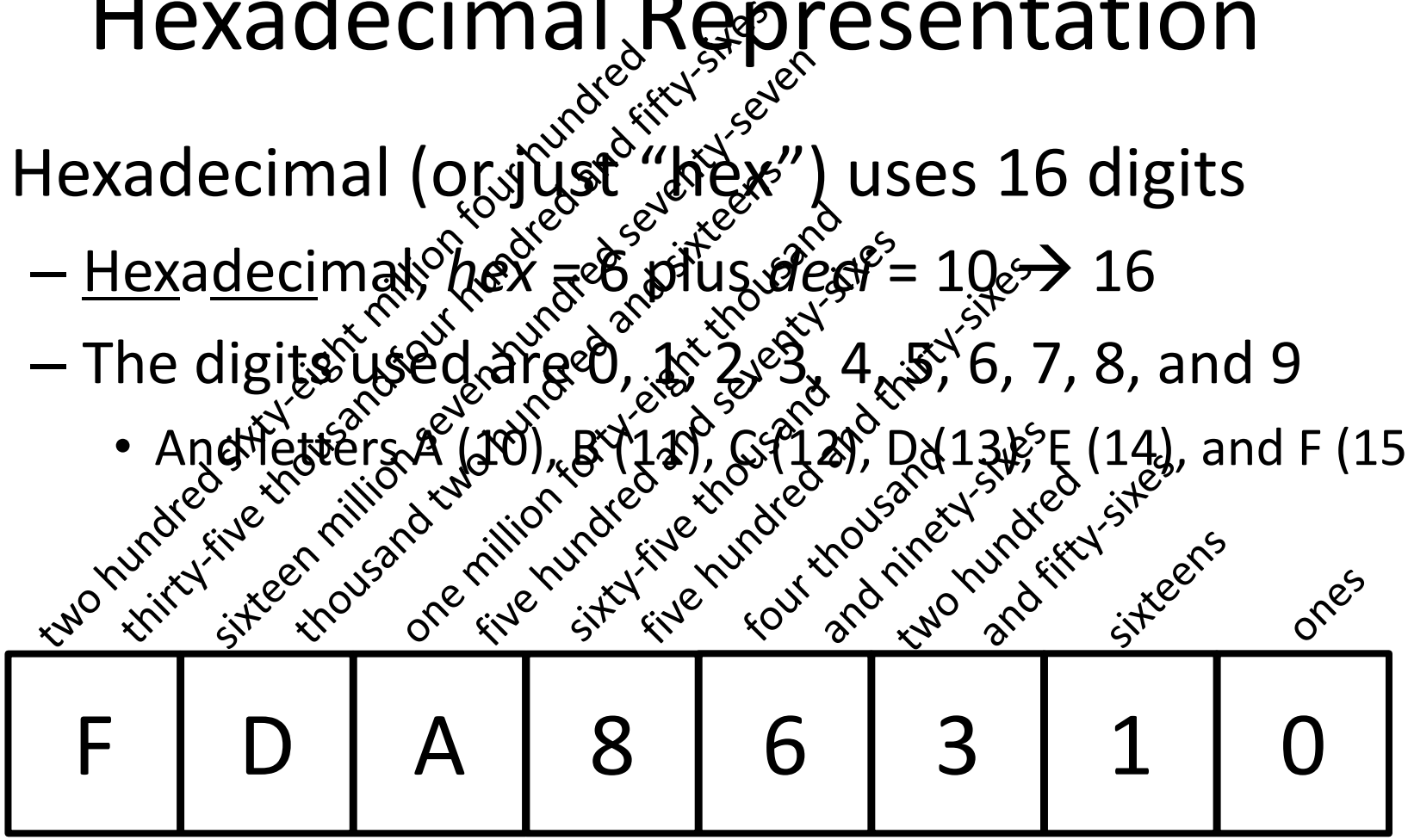


Binary Representation

- Binary uses 2 digits
  - Binary, *bi* = 2
  - The digits used are 0 and 1

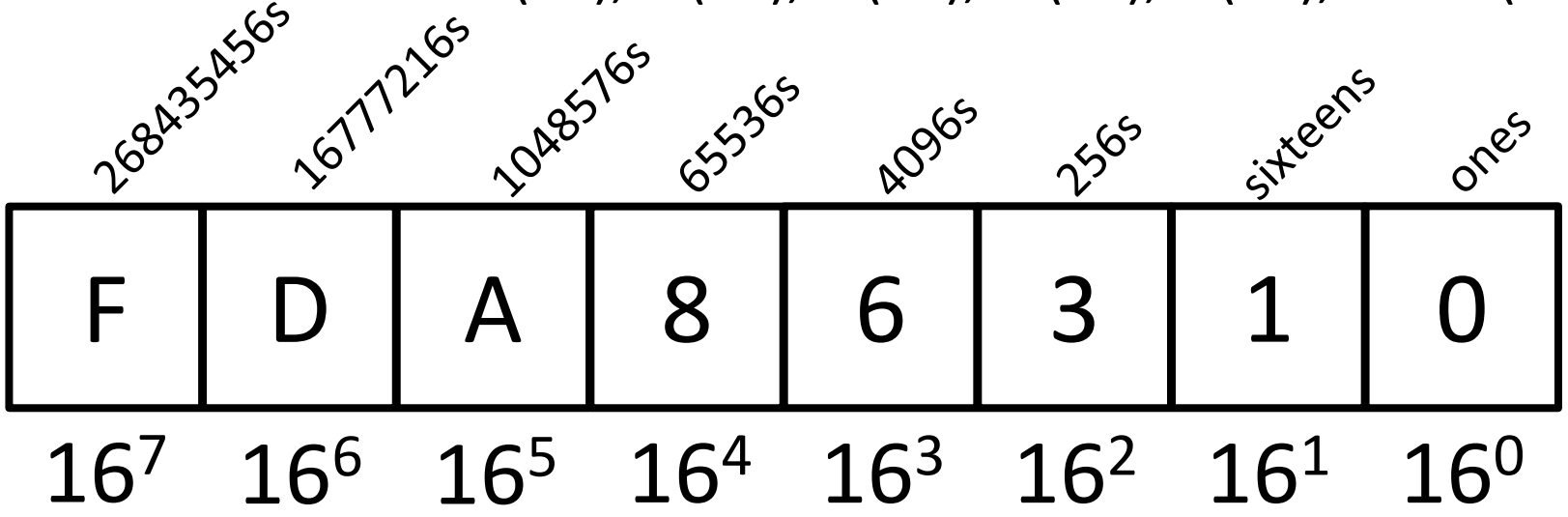
Hexadecimal Representation

- Hexadecimal (or just "hex") uses 16 digits
  - Hexadecimal, *hex* = 6 plus *decem* = 10 → 16
  - The digits used are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
    - And letters A (10), B (11), C (12), D (13), E (14), and F (15)



Hexadecimal Representation

- Hexadecimal (or just "hex") uses 16 digits
  - Hexadecimal, *hex* = 6 plus *deci* = 10 → 16
  - The digits used are 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
    - And letters A (10), B (11), C (12), D (13), E (14), and F (15)



Hex to Binary Conversion

- A hexadecimal digit can be easily represented as four digits of binary (with leading zeros)

Hex	Binary	Hex	Binary	Hex	Binary	Hex	Binary
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

- This makes conversion very simple
  - **7A0F** becomes **0111 1010 0000 1111**
  - **1100 0010 0110 1001** becomes **C269**

Hex to Decimal Conversion

- Possible to convert between decimal and hex
  - But it requires calculating out multiples of 16
- Simpler to make a "side trip" to binary as an in-between step when converting
  - 240 becomes **`1111 0000`** becomes **`F0`**
    - **`F0`** is equal to $(15 * 16^1) + (0 * 16^0) = 240 + 0 = 240$
  - **`7D`** becomes **`0111 1101`** becomes 125
    - **`7D`** is equal to $(7 * 16^1) + (13 * 16^0) = 112 + 13 = 125$

Number System Notation

- Because number systems share a subset of the same digits, it may be confusing which is which
  - For example, what is the value of 10?
    - In decimal it's 10, in binary it's 2, and in hex it's 16

- To prevent this, numbers may often be prefixed with **`0b`**, **`0d`**, or **`0x`** (binary, decimal, hex):
  - **`0b1100`** is binary, and has a value of 12
  - **`0x15`** is hexadecimal, and has a value of 21

ASCII Values

ASCII Values

- ASCII is how text is represented in computers
  - Just like binary is how numbers are represented
- In ASCII, every character has a unique, individual numerical code
  - Lowercase and uppercase characters are separate
  - Codes go from 0 to 127
    - Why 127?

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

A TABLE

"control" characters

uppercase letters

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

symbols & numbers

lowercase letters

Comparing Strings

- The values of the ASCII characters are used when comparing strings together
  - Which can lead to some "weird" results

```
>>> "cat" < "dog"
True
>>> "cat" < "Dog"
False
>>> "DOG" < "dog"
True
```

?

More on Comparing Strings

- Gets even more complex when you start adding in numbers and symbols

```
>>> "2" < "one"
True
>>> "good?" < "good!"
False
>>> "UK" < "U.K."
False
```

Rules for Comparisons

- To avoid (some) of these issues:

- Always use `.lower()` for comparing strings
- Pay attention to symbols
  - *e.g.*, spaces, hyphens, punctuation, etc.
  - Either remove them, or keep them in mind as part of the order

ASCII Characters to ASCII Values

- We can convert between ASCII characters and their values using **`ord()`** and **`chr()`**

- The **`ord()`** function takes in a single character, and returns its ASCII value
- The **`chr()`** function takes in an integer, and returns its ASCII character

Using `chr()` and `ord()`

```
>>> chr(65)
'A'
>>> chr(65+32)
'a'
>>> ord('?')
63
>>> ord("d")
100
>>> ord("e")
101
```

Project 3

Project 3 Tips

- Hopefully you have started by now!
  - Work on it a little everyday
- You have been given some solved puzzles
  - Which means you don't need a working `solve()` to test the other parts of your project
    - Just load in the solution from the file
- Solve the puzzle once, and store the solved puzzle to use it later in your code
  - Don't resolve it every time you need it
- Make your own puzzles to test!

Project 3 and Deep Copy

- You will need to make a deep copy of the 2D list used to hold your Sudoku board
  - Simply using `new = old[:]` will not work
- We recommend making a function to do this
  - Test that your function works before using it

- Do ***NOT*** use the built-in `deepcopy()` function, or you will lose major points!!!

Do Not Cheat on Project 3

- Yes, this project has solutions on the internet
  - Yes, we have copies of all of them
  - Yes, we will go looking for new ones after it's due
- Yes, you could pay someone else to do it
  - Yes, we know of the sites where you can get this done
  - Yes, we will spot "elegant" code that you didn't write
- Yes, there are libraries to deep copy in python
  - Yes, you will get points off for using them
  - You should not be importing anything for this project

Questions?

Announcements

- Project 3
  - Design is due Tuesday, December 4th
  - Project is due Tuesday, December 11th

- Final exam is when?
  - Friday, December 14th from 6 to 8 PM
  - Locations will be posted on the course website
  - Common final