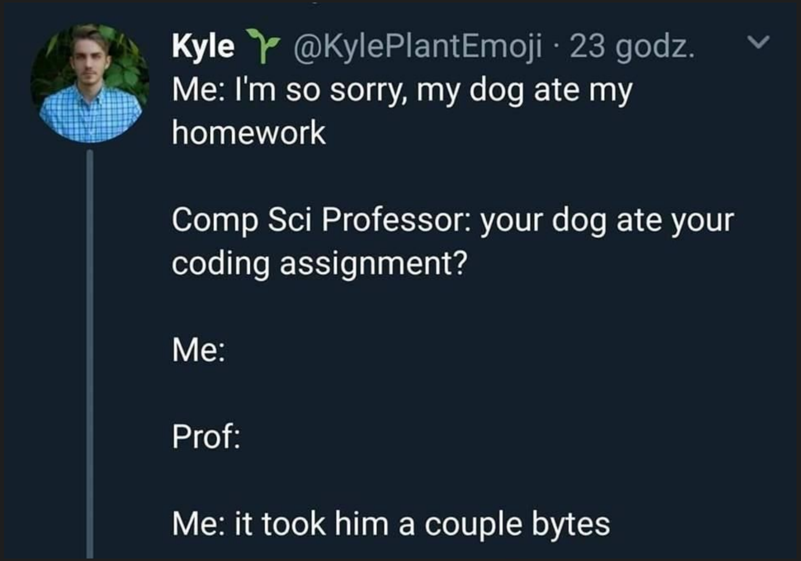

Kyle @KylePlantEmoji · 23 godz.

Me: I'm so sorry, my dog ate my homework

Comp Sci Professor: your dog ate your coding assignment?

Me:

Prof:

Me: it took him a couple bytes

comp1511 week 7

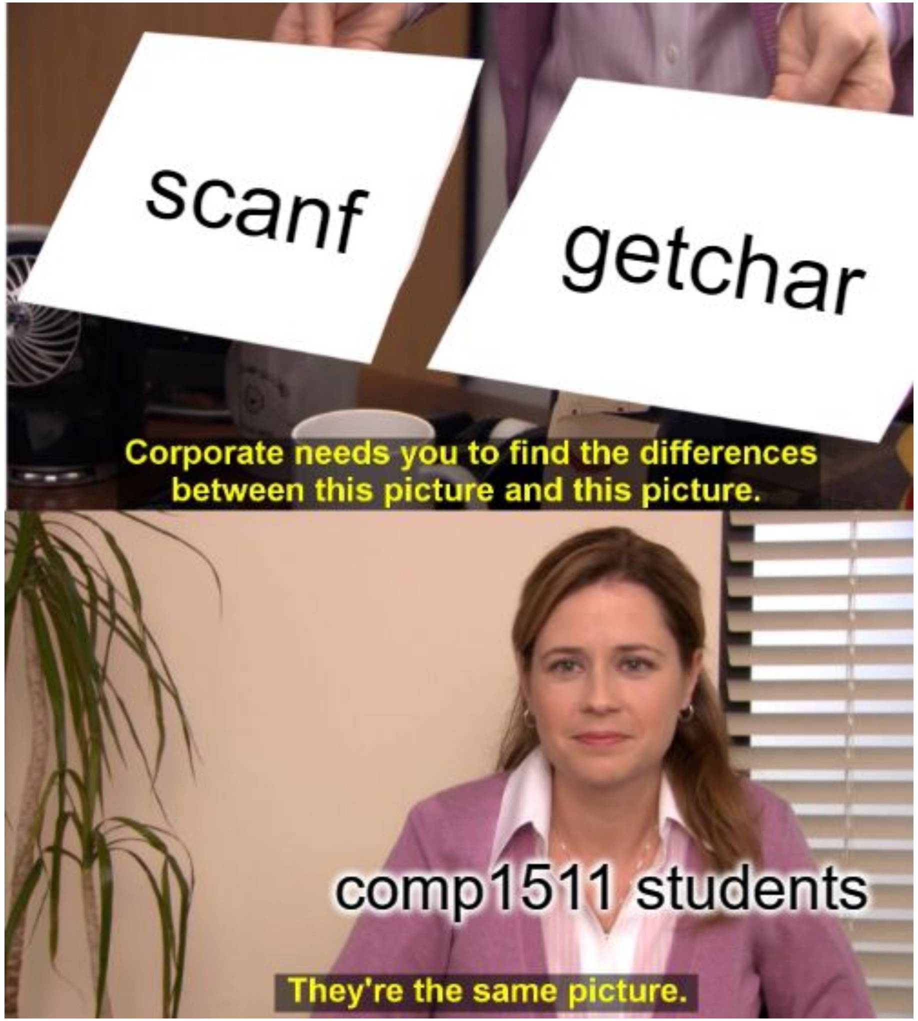
starting ~9:08am

notices

- assignment 1 is almost done!
    - congratulations :D

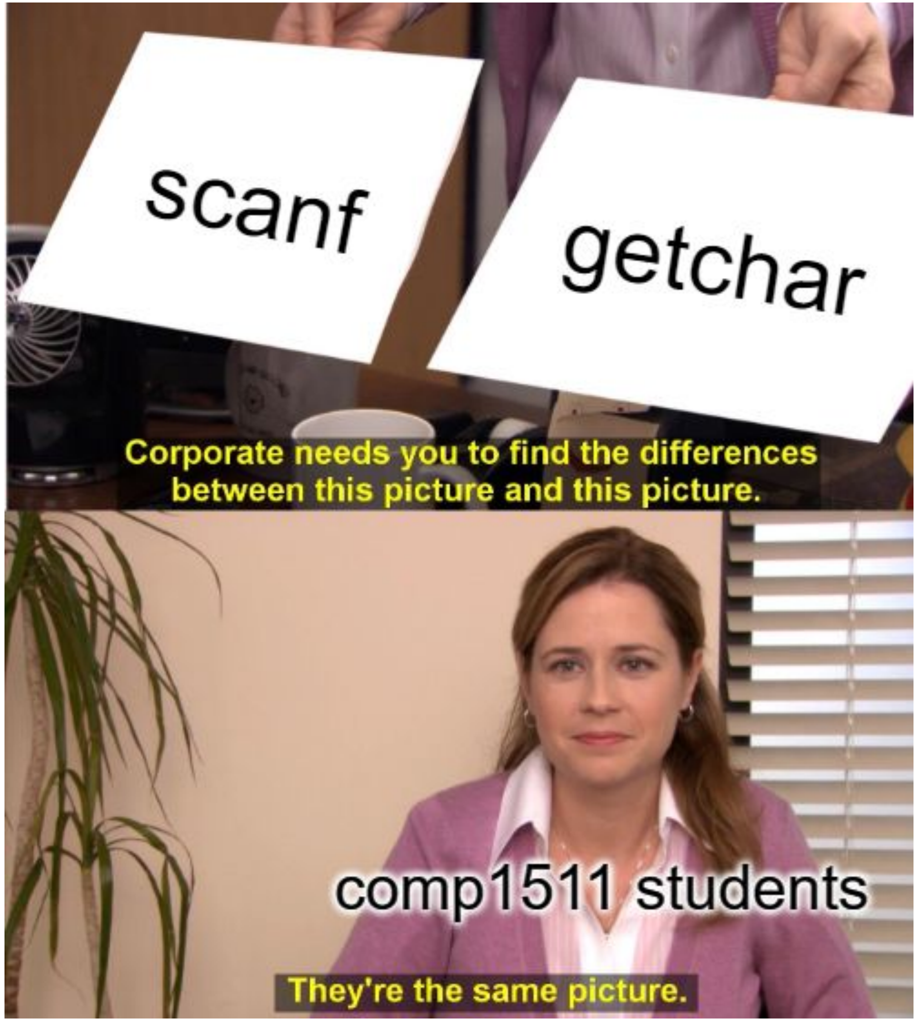
today

- getchar() and putchar()
- strings
- fgets
- command line arguments
- struct pointers
- bonus crypto stuff if you’re interested


scanf
getchar
Corporate needs you to find the differences between this picture and this picture.
comp1511 students
They're the same picture.

scanf()

- returns the number of items successfully read in
- what's the return value of:
  - an invalid input?
  - end of file? (CTRL-D)


scanf
getchar
Corporate needs you to find the differences between this picture and this picture.
comp1511 students
They're the same picture.

scanf()

- returns the number of items successfully read in
- what's the return value of:
  - an invalid input?
  - end of file? (CTRL-D)

getchar()

- returns the character it scans in OR EOF if end of input
- aka return an **int** not a **char**

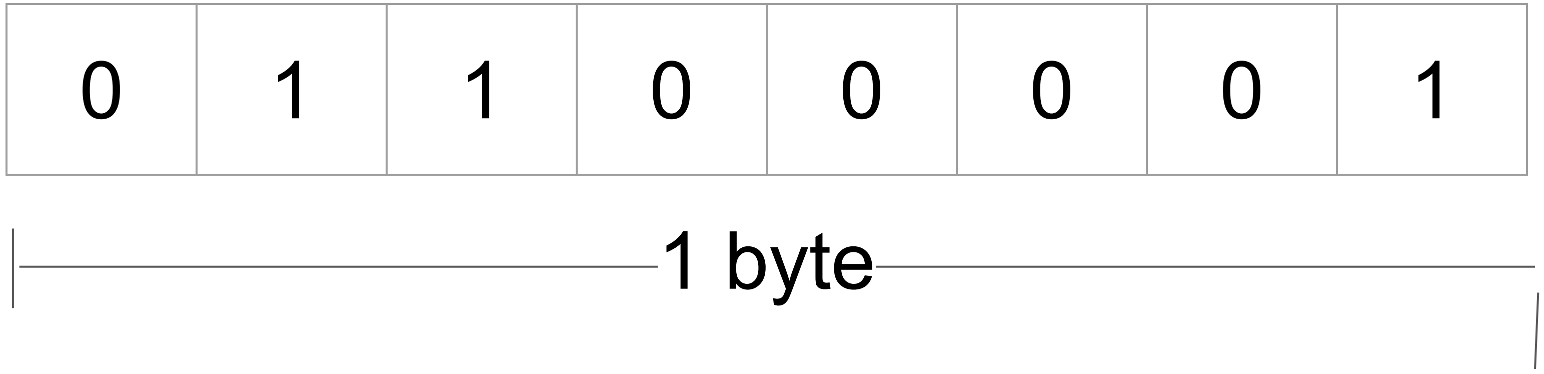
getchar() and putchar()

this	**is very similar to**

int ch;
ch = getchar();
	char ch;
	scanf(“%c”, &ch);

putchar(ch);	printf(“%c”, ch);

what is a char?



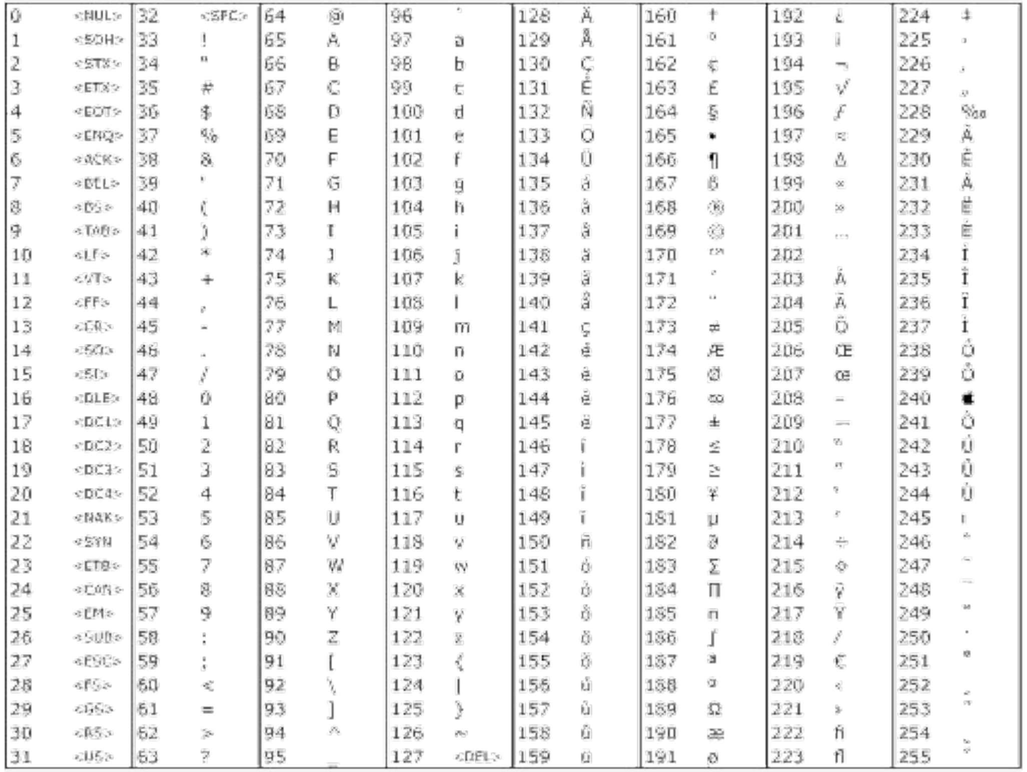
7. Write a program `sum_digits.c` which reads characters from its input. When the end of input is reached it should print a count of the number of digits in its input and their sum.

   The only functions you can use are `getchar()` and `printf()`.

   For example:

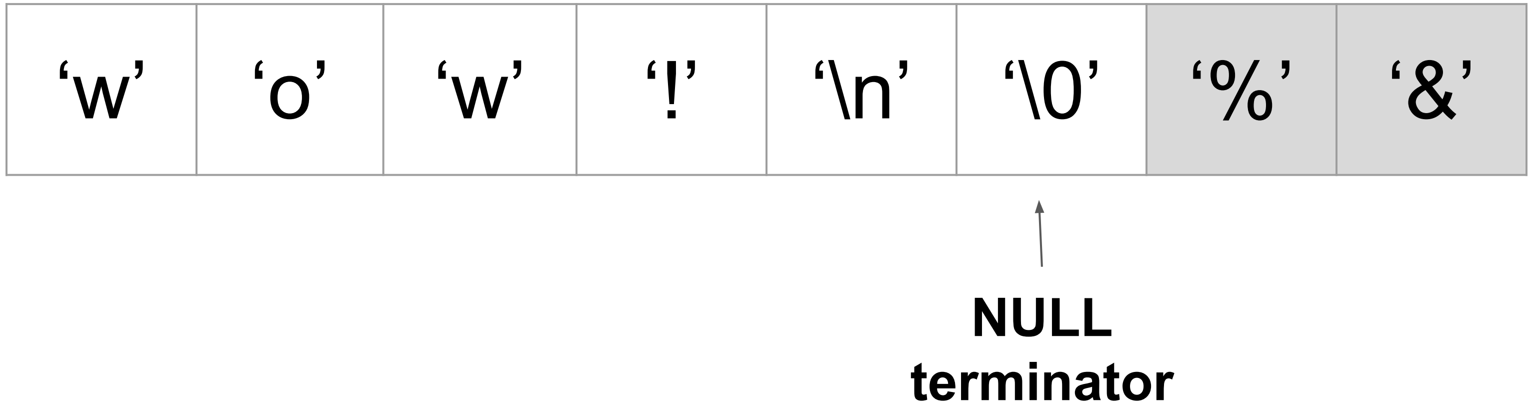
```
$ ./sum_digits
1 2 3 o'clock
4 o'clock rock
Ctrl-D
Input contained 4 digits which summed to 10
$ ./sum_digits
12 twelve 24 twenty four
thirty six 36
Ctrl-D
Input contained 6 digits which summed to 18
```

ascii table



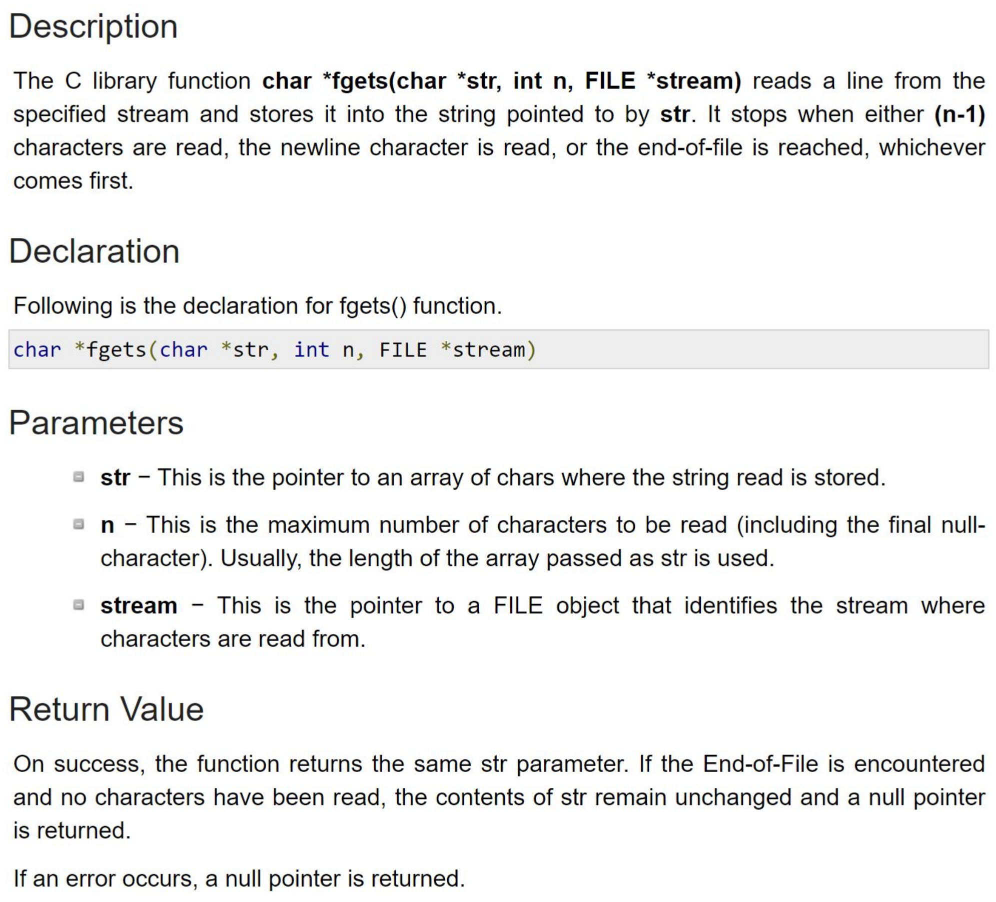
0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	∆	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	“	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	”	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	‘	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	’	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	∑	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	∏	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	⁄	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	˚
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	ﬁ	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	ﬂ	255	ˇ

strings



```c
int secret_function(char *word) {
    int i = 0;
    int result = 0;
    while (word[i] != '\0') {
        if (word[i] >= 'a' && word[i] <= 'z') {
            result++;
        }
        i++;
    }
    return result;
}
```

fgets



Description

The C library function **char *fgets(char *str, int n, FILE *stream)** reads a line from the specified stream and stores it into the string pointed to by **str**. It stops when either **(n-1)** characters are read, the newline character is read, or the end-of-file is reached, whichever comes first.

Declaration

Following is the declaration for fgets() function.

```
char *fgets(char *str, int n, FILE *stream)
```

Parameters

- **str** − This is the pointer to an array of chars where the string read is stored.
- **n** − This is the maximum number of characters to be read (including the final null-character). Usually, the length of the array passed as str is used.
- **stream** − This is the pointer to a FILE object that identifies the stream where characters are read from.

Return Value

On success, the function returns the same str parameter. If the End-of-File is encountered and no characters have been read, the contents of str remain unchanged and a null pointer is returned.

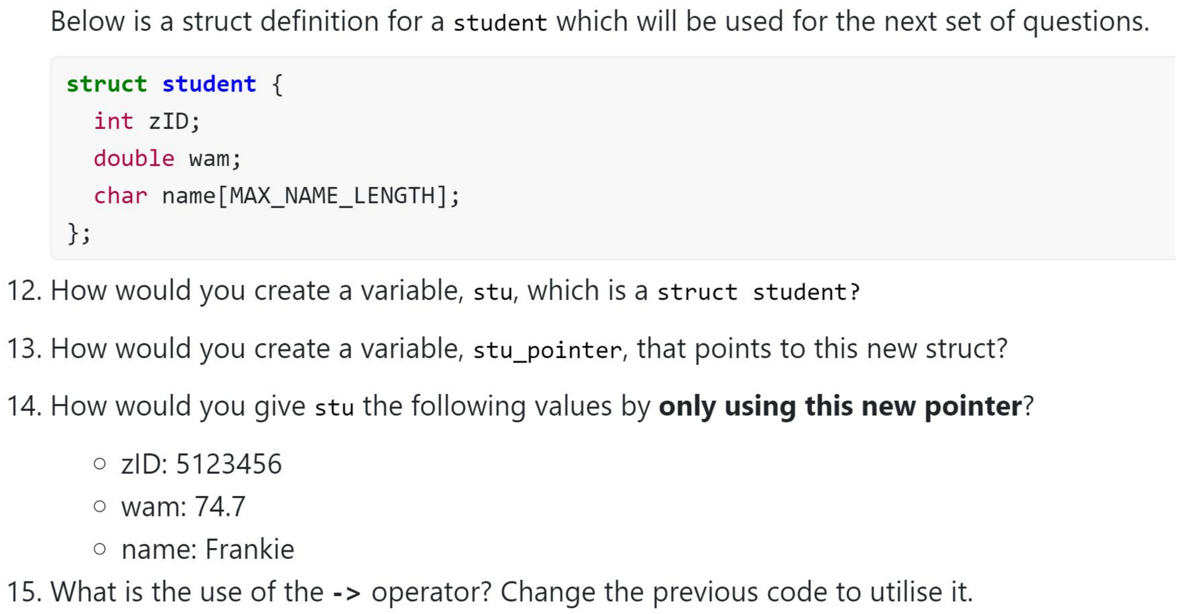
If an error occurs, a null pointer is returned.

https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm

what is stored in argc and argv?

```
./add 10 20 30
```

struct pointers



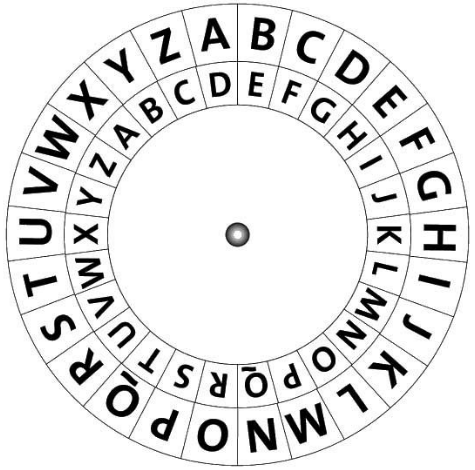
Below is a struct definition for a `student` which will be used for the next set of questions.

```c
struct student {
  int zID;
  double wam;
  char name[MAX_NAME_LENGTH];
};
```

12. How would you create a variable, `stu`, which is a `struct student?`
13. How would you create a variable, `stu_pointer`, that points to this new struct?
14. How would you give `stu` the following values by **only using this new pointer**?
    - zID: 5123456
    - wam: 74.7
    - name: Frankie
15. What is the use of the `->` operator? Change the previous code to utilise it.

intro to crypto
(bonus slides)

caesar cipher



cyclically shift each letter *k* places forward

k = 3

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

For *k* = 3, the plaintext HELLO is encrypted as KHOOR

simple substitution cipher

permute the alphabet for a key, then map letters to encrypt.

mapped alphabet to a scrambled version

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
P	Q	S	T	U	V	W	X	Y	Z	C	O	D	E	B	R	A	K	I	N	G	F	H	J	L	M

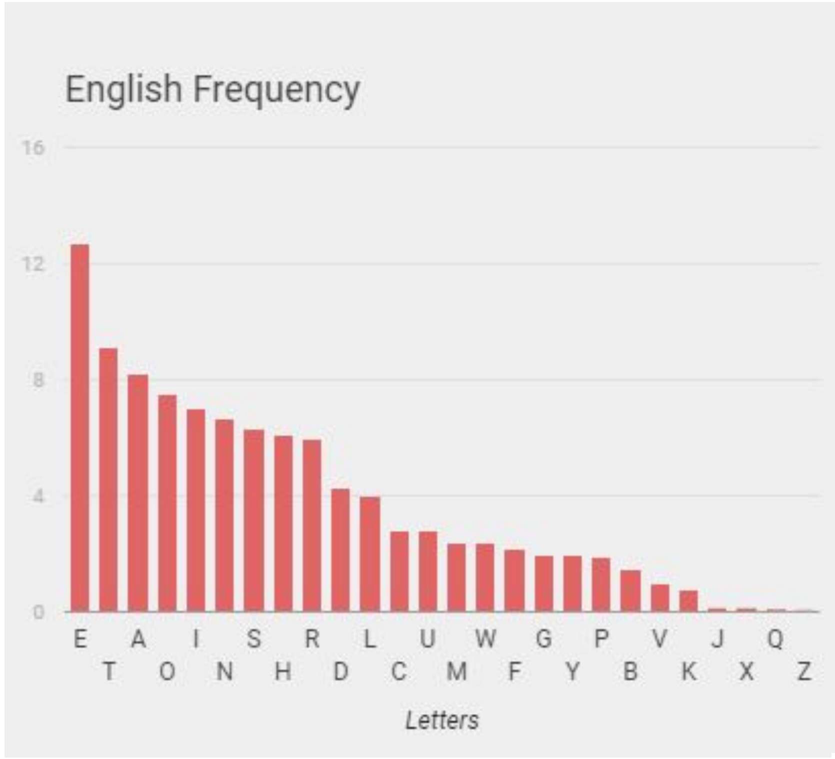
The plaintext HELLO is encrypted as XUOOB

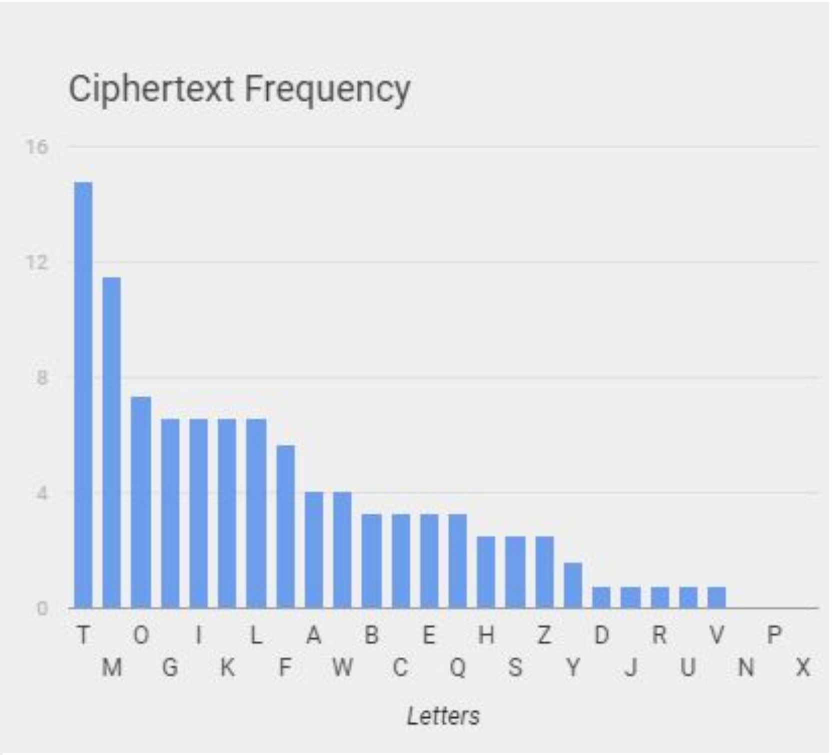
number of keys

$$|K| = 26! \approx 4 \times 10^{26}$$

that’s a big number!!!

decryption - the magic of frequency


English Frequency
E T A O I N S H R D L C U M W F G Y P B V K J X Q Z
Letters


Ciphertext Frequency
T M O G I K L F A W B C E Q H S Z Y D J R U V N P X
Letters