1. What does SHA256 do? (The function)

2. The hash of my password starts 04A12BC9. (nothing)
   What do you know about it?

3. Write a main function that prints the first command-line argument.

1 SHA256 takes:

- an input char array (any size)
- a length
- an output array (32 bytes available)

and changes the output array to have the SHA256 hash of the first length bytes of input

args.c

3.
```
int main(int argc , char** args) {
    ~~args~~
    char* first_arg = args[1];
    printf("%s", first_arg);
}
```

```
$gcc args.c -o args
$ ./args banana apple
banana $
```

args[0] => "./args"
args[1] => "banana"
⋮

0x..10	"./args"
0x..20	"banana"
args	0x....A0
0xA0	0x....10 0x....20 0x...30

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

// char *strtok(char *str, const char *delim);
// Returns a pointer to the next token in str, delimited by delim.
// First call: pass the string. Subsequent calls: pass NULL.
// Replaces delim with '\0'returns pointer into original string.

int main() {
  char str[] = "Joe Politz,jpolitz@ucsd.edu,Instructor";
  printf("str@%p: %p \"%s\"\n", &str, str, str);

  char* a = strtok(str, ",");          ~ strtok remembered 0x...5B
  char* b = strtok(NULL, ",");         ~ strtok remembered 0x...6C
  char* c = strtok(NULL, ",");         ~ strtok remembered 0x..77
  char* d = strtok(NULL, ",");

  printf("a@%p: %p \"%s\"\n", &a, a, a);
  printf("b@%p: %p \"%s\"\n", &b, b, b);
  printf("c@%p: %p \"%s\"\n", &c, c, c);
  printf("d@%p: %p\n", &d, d);

  printf("\nstr after strtok:\n");
  for (int i = 0; i < 39; i++) {
    if (i % 8 == 0) printf("%p:", &str[i]);
    if (str[i]) printf(" '%c'", str[i]);
    else printf(" NUL");
    if (i % 8 == 7) printf("\n");
  }
  printf("\n");
}
```

```
$ gcc strtok.c -o strtok
$ ./strtok
str@0x16b7b6e51: 0x16b7b6e51 "Joe Politz,jpolitz@ucsd.edu,Instructor"
a@0x16b7b6e40: 0x16b7b6e51 "Joe Politz"
b@0x16b7b6e38: 0x16b7b6e5c "jpolitz@ucsd.edu"
c@0x16b7b6e30: 0x16b7b6e6d "Instructor"
d@0x16b7b6e28: 0x0

str after strtok:
0x16b7b6e51: 'J' 'o' 'e' ' ' 'P' 'o' 'l' 'i'
0x16b7b6e59: 't' 'z' NUL 'j' 'p' 'o' 'l' 'i'
0x16b7b6e61: 't' 'z' '@' 'u' 'c' 's' 'd' '.'
0x16b7b6e69: 'e' 'd' 'u' NUL 'I' 'n' 's' 't'
0x16b7b6e71: 'r' 'u' 'c' 't' 'o' 'r' NUL
```

Variable/Role	**Address**	**Data**								
	0x...00									
	0x...08									
	0x...10									
	0x...18									
	0x...20									
d	0x...28	0x0 (NULL)								
c	0x...30	0x16b7b6e6d								interior pointer
b	0x...38	0x16b7b6e5c								interior pointer
a	0x...40	0x16b7b6e51								
	0x...48									
str	0x...50		J	o	e	␣	P	o	l	
	0x...58	i	t	z	~~,~~\0	j	p	o	l	
	0x...60	i	t	z	@	u	c	s	d	
	0x...68	.	e	d	u	~~,~~\0	I	n	s	
	0x...70	t	r	u	c	t	o	r	\0	
	0x...78									
	0x...80									
	0x...88									
	0x...90									
	0x...98									
	0x...A0									
	0x...A8									
	0x...B0									
	0x...B8									
	0x...C0									
	0x...C8									
	0x...D0									
	0x...D8									
	0x...E0									
	0x...E8									
	0x...F0									
	0x...F8									

How to implement a shell. You've been using "bash"

on mac — "zsh"

Joe likes - "fish"

windows - "PowerShell"

1. Prints a prompt
2. "Parses" a command + args the user typed
3. Runs that command (by asking the OS)

plus bonuss

- tab complete
- saves history
  (up arrow, Ctrl-R)
- I/O redirection

```
$ ./prog <input >output
```