C - STRING FUNCTIONS

CONCATENATION

- `strcat(char *s1, char *s2)`
  - appends copy of `s2` to end of `s1`
  - `s1` must have enough space (or else ...)
- `strncat(char *s1, char *s2, size_t n)`
  - safer -- will not append more than `n` characters, then adds terminating char

COPYING

- `strcpy(char *dest, char *src)`
  - copies `src` into `dest`
  - copies terminating '\0' as well
  - make sure `dest` has enough space
- `strncpy(char *dest, char *src, size_t len)`
  - copy at most `len` chars from `src` into `dest`
  - `src` less than `len`, fills with '\0'
  - otherwise, does not add '\0'

LENGTH

- `strlen(char *s)`
  - returns number of characters preceding '\0'
  - the string needs to be null terminated
- `strnlen(char *s, size_t maxlen)`
  - returns either result of `strlen()` or `maxlen`, whatever is smaller

COMPARING

- `int strcmp(char *s1, char *s2)`
  - returns positive integer if `s1` > `s2`
  - returns 0 if `s1` == `s2`
  - returns negative integer if `s1` < `s2`
- `int strncmp(char *s1, char *s2, size_t n)`
  - same, but only comparses up to `n` characters
  - any characters after a '\0' not compared

SEARCHING/INDEX OF

- `char* strchr(char *s, int c)`
  - returns pointer to first occurence of `c` in `s`
  - returns NULL if not found
- `char* strrchr(char *s, int c)`
  - returns pointer to last occurence of `c` in `s`
  - returns NULL if not found

SEARCHING/INDEX OF (CONT.)

- `char* strstr(char *haystack, char *needle)`
  - finds first occurence of substring `needle` in `haystack`
  - does not compare '\0'
  - returns pointer to beginning of substring in `haystack`
  - returns NULL if not found

MEMORY MOVING / SETTING

- `memcpy(void *dest, void *src, size_t n)`
  - copies n bytes from `src` into `dest`
- `memmove(void * dest, void *src, size_t n)`
  - copies n bytes from memory are `src` to `dest`
  - allows memory areas to overlap
- `memset(void *s, int c, size_t n)`
  - fills first `n` bytes of `s` with byte `c`