Strings!

Important testimony:

- “Understanding strings is critical for all students in CS 136.” – Abraham Lincoln.
- “I wish I had paid more attention to strings and dynamic memory management.” – Albert Einstein.
- ‘This is not a valid string because it uses the wrong quotation marks.’ – Bill Gates

Strings Review

- A string in C is an array of characters that is NULL terminated.

For example:

```c
char s1[] = {'s', 't', 'r', 'i', 'n', 'g'}  // s1 is not a string
char s2[] = {'s', 't', 'r', 'i', 'n', 'g', '\0'}  // s2 is a valid string
const char * s3 = "string";  // s3 is also a valid string
```

Useful <string.h> functions

```c
int strlen(char s[])  // finds the length of the string s,  
// starting from s[0] and ending when it sees '\0',  
// not counting '\0' itself.

int strcmp(char s1[], char s2[])  // compares s1 and s2 lexicographically  
// If both strings are identical, it returns 0  
// If str1 comes before str2, it returns an int < 0  
// if str2 comes before str1, it returns an int > 0
```

Useful <string.h> functions

Never put strlen or other $O(n)$ functions in a loop condition!
Beginners often make this mistake, which often causes increases to runtimes.

```c
char * strcpy(char * dest, const char * src)
// overwrites the contents of dest
// with the contents of src

char * strcat(char * dest, const char * src)
// copies (appends or concatenates) src
// to the end of dest
```

Exercise: Substrings n’ Things
Implement the following function using pointer arithmetic (do not use array notation):
```
// substr: returns a pointer to the first occurrence of
// sub inside str. if no match is found, returns NULL.
// this is suspiciously similar to "strstr".
// requires: str is a valid, non-NULL string. sub is a
// valid, non-NULL string.
char* substr(const char* str, const char* sub);
```

Exercise: Duplicate aaaaaaaa Character
Implement the following function:
```
// duplicate: returns a dynamically allocated,
// NULL-terminated string containing n copies of
// character c. on malloc failure, returns NULL.
// requires: n > 0.
// effects: dynamically allocates the returned string.
char* duplicate(char c, int n);
```