Arrays/Pointers
Variables review
int a[4];
char * b[] = {"roll", "tide"};
double c[2] = {4.0, 9.7};
Accessing Parts of an Array

Use indexing!

\[ a[0] \]

To change value at index:

\[ a[3] = 7; \]
Remember: counting starts at zero!
Be careful!
Accessing beyond the bounds of an array will not give you an error!
arrays.c
Aliases

double a[] = {1.4, 3.9};

    double *b = a;

    b[0] = 2.2;
To create alias, add a *!
double **b;

b[0] = a;
• lvalues: things on the left side of an assignment statement, we update their values

• lvalues refer to memory locations

• rvalues: things on the right side of an assignment statement, we use their values
Size of arrays

sizeof(a) : returns number of bytes allocated

sizeof(a) / sizeof(double)

sizeof(a) / sizeof(*a)

Does not work for pointers!
More Indexing

• Can also index with pointers
• Dereferencing: \( *a = 0; \) same as \( a[0] = 0; \)
Static vs. Dynamic

- Previous arrays were statically allocated
- Dynamically allocated arrays last for length of your program
Strings

//test
char *s1 = "hello";
char s2[6] = { 'h', 'e', 'l', 'l', 'o', '\0' };
printf("%s\n", s1);
printf("%s\n", s2);

string array must end with null character
Changing a String

```c
char *r = "rat";  // r points to a literal string
char s[] = { 'r', 'a', 't', '\0' };  // s pseudopoints to an array of char
char *t = r;     // t points to a literal string
char *u = s;     // u points to an array of char

r[0] = 'c';       // BAD
s[0] = 'c';       // OK
t[0] = 'c';       // BAD
u[0] = 'c';       // OK
```
Length of a String

\texttt{strlen(s1)}

\texttt{strlen(s2)}
Copying a String

char *a = “roll tide”;

char b[10]; // don’t forget null character!

strncpy(a, b, 10);
Comparing Strings

`strcmp(a, b);`

Returns 0 if same, negative int if first string comes first, positive int if second string comes first
2048strings.c