C - POINTERS, ARRAYS

ALLOCATING MEMORY

- `malloc(size)` -> request a certain number of bytes of memory
  - on the heap
  - contiguous
  - returns address of start of block
  - needs to be cast to appropriate pointer
- memory for one int:

```
int *p;
p = (int *) malloc(sizeof(int));
```

POINTERS AND ARRAYS

- Array name = pointer to element 0 that can't be changed
- Pointer arithmetic becomes useful
  - walking down array

ALLOCATING MORE MEMORY

- Use `malloc`, but request more bytes
- Often used to malloc memory for an array

```
int *p;
p = (int *) malloc(sizeof(int)*100);
```

ALLOCATING MEMORY (CONT.)

- `calloc(count, size)`
  - `count` -> number of items
  - `size` -> size of item
  - allocated contiguously
  - initialized to 0

```
int *p;
p = (int *) calloc(100, sizeof(int));
```

ALLOCATING MEMORY (CONT.)

- `realloc(ptr, size)`
  - `ptr` -> current pointer to memory trying to resize
  - `size` -> overall size wanted
- may not be enough room to enlarge
  - creates new, copies to new, frees the old
  - additional not guaranteed to be 0 filled

```
int *p = (int *) malloc(sizeof(int)*10);
p = (int *) realloc(ptr, sizeof(int)*20);
```

FREEING MEMORY

- `free(ptr)`

```
int *p = (int *) malloc(sizeof(int)*10);
free(p);
```