# Dynamic memory allocation

- memory allocated at run-time
- allocated at the "heap"
- CTA (Compile time allocation)
  ↳ memory for variables is allocated by the compiler at run time
  ↳ reqires exact size & type of storage
- DMA = calculates & allocates memory as it runs.
- heles varables last beyond the lifetime of its current scope

when they are dynamacally allocated, they can be accessed beyond the current scope

- sometimes, we don't know how much memory we need, which is us dynamic allocaton is great

malloc, calloc, & reallloc are all used to allocate memory
↳ They return a pointr to the allocated memory

- the memory is free using the free funcyon
  ↳ If not freed Propery (aka FILA) they can lead to memory leaks

The heap

- unmaged memory
- basically no limitations, but your computers limits
- it takes more effort and time to access because it needs pointrs

malloc

- returns a pointer to bytes of Uninitiaized memory allocated on the heap
-▷ Sometimes the data can be junk
-▷ don't try to allocate 0

doesn't check for overflow of size

calloc

- denotes # of objects & size of each object
- returns a pointer, each byte has been initialized to 0
- Contents of the allocated memory are known

Realloc

reallocates pointer to a new point

↳▷ deallocates the old object pointed to by the pointer & returns a pointer to bytes of allocated

Stack

"Struct" -> Stack

Uses a [illegible]

↳ size, top, & entries

Free

- deallocates the memory space pointed by the ponter
- if you don't free memory, you create a leak
- if you try to acess memory that doesn't exist or you have no access to, you'll get a core dump
- set pointer to Nil to mitigate use-after-free vulnerabilites

Debugsing!

gdb, linfer, valgrind are all used to fix core dump errors

Static vs dynamic analyzers

Static (infer): analyze the source code before it runs

- compared to rules

Dynamic (valgrind) - track errors that occrs during program execution

↳ things that don't occur during execution arent analyzed