C - STACK VS HEAP

STACK VS HEAP

- Two different areas of memory
- Typically implemented with stack and heap data structures, respectively
- How a variable is declared -> where it goes
- Everything we've looked at so far is on stack

STACK

- New block whenever a function called
- Block = scope
    - local variables stored in that block
    - block freed when function returns (aka variables disappear)

HEAP

- Memory allocated dynamically
- Memory can be allocated and freed at any time
- Entire program has access to the heap (no real scope)
- Memory must be explicitly freed
- Memory leaks
    - Lose access to memory
    - Computer doesn't know memory is no longer accessible

WHEN TO USE DYNAMIC (HEAP)?

- Need a lot of memory
    - Stack is typically fairly limited in size
    - Request too much -> program crashes
- Needs to persist after function returns
    - Ex: array made in function that still need access to in main
- If amount of memory needed unknown / changes
    - Ex: array size not determined in advance
    - Ex: array that needs to change size
- Safest to use if size is not known at compile time
    - VLAs (C99) allow, but as of C11 it's optional

MEMORY ALLOCATION

- Two types:
  - compile time memory allocation
    - stack
    - Ex: `int a;`,`int arr[20];`
  - runtime (dynamic) memory allocation
    - heap
    - Ex: anything using `malloc()`, `calloc()`,`realloc()`