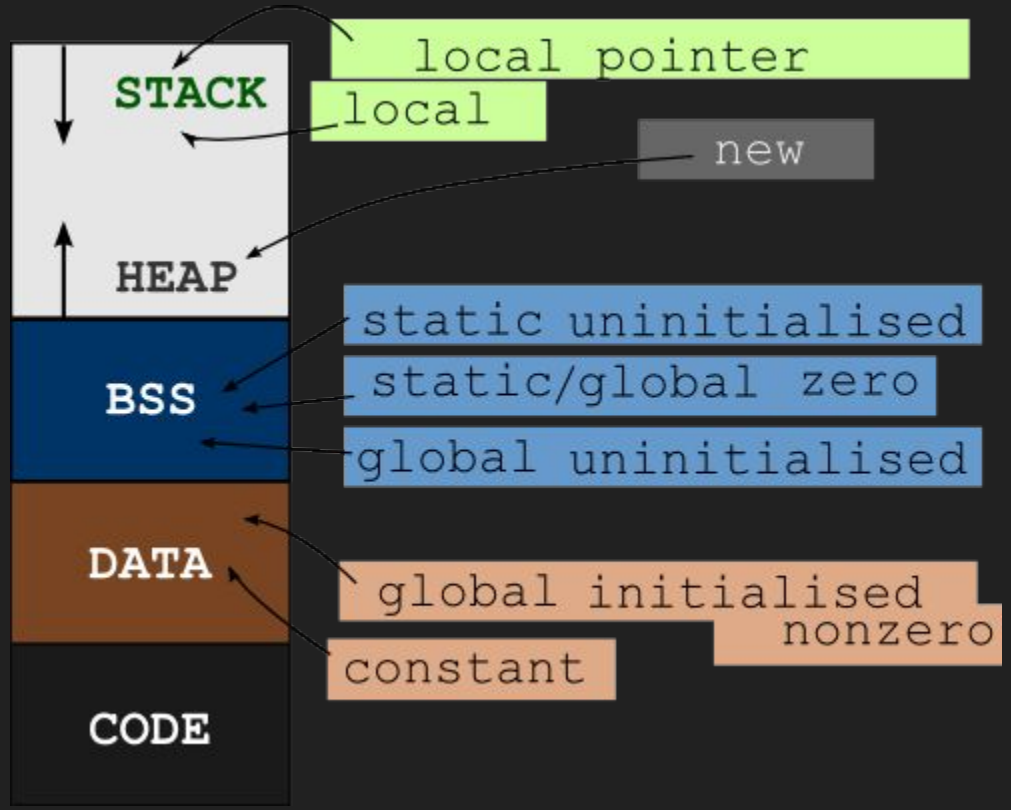
C and the HEAP

Derrick

What is the Head?

- An executing program is divided into four parts:
  - Stack: Provides storage for local variables, alters size as the program executes
  - Data Segment: global variables and strings stored here. Fixed size.
  - Code Segment: your source code, functions `main, printf, scanf` etc. Read only. Fixed size.
  - Heap: “Dynamic memory” the heap is available for us to use and may change size as the program executes.

Memory Layout (segmentations) **(optional)**



Dynamic Arrays

- Arrays in C have a fundamental problem - their size must be fixed when the program is written
- There is no way to increase (or decrease) the size of an array once the program is compiled
- Dynamic arrays are different, their size is fixed at run time and may be changed as often as required
- Only a pointer is required

Steps for Dynamic Arrays

- Declare a pointer to the type of the array elements.
- Initialize the pointer via `calloc` or `malloc` using the total storage required for the array.
- Check the pointer if it's `NULL`.
- Increase or decrease the number of elements by calling the `realloc` function.
- Release the storage by calling `free.`

malloc, calloc, realloc <stdlib.h>

Function	Use of Function
malloc()	Allocates requested size of bytes and returns a pointer first byte of allocated space
calloc()	Allocates space for an array elements, initializes to zero and then returns a pointer to memory
free()	deallocate the previously allocated space
realloc()	Change the size of previously allocated space

Example

Demo

`realloc` can do it all

- The `malloc` and `free` can be replaced with `realloc`

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

P = realloc(NULL, s * sizeof(int));

free(p);

realloc(p, 0);
```

Allocating dynamic Arrays of Arrays

- Be careful when dealing with arrays of arrays

```
int **dynamic_2d_array(int row, int col)
{
    int **arrs = (int **) malloc(row * sizeof(int *));
    for (int i = 0; i < row; i++)
        arrs[i] = (int *) malloc(col * sizeof(int));
    return arrs;
} /* left out error checks for malloc for space */
```