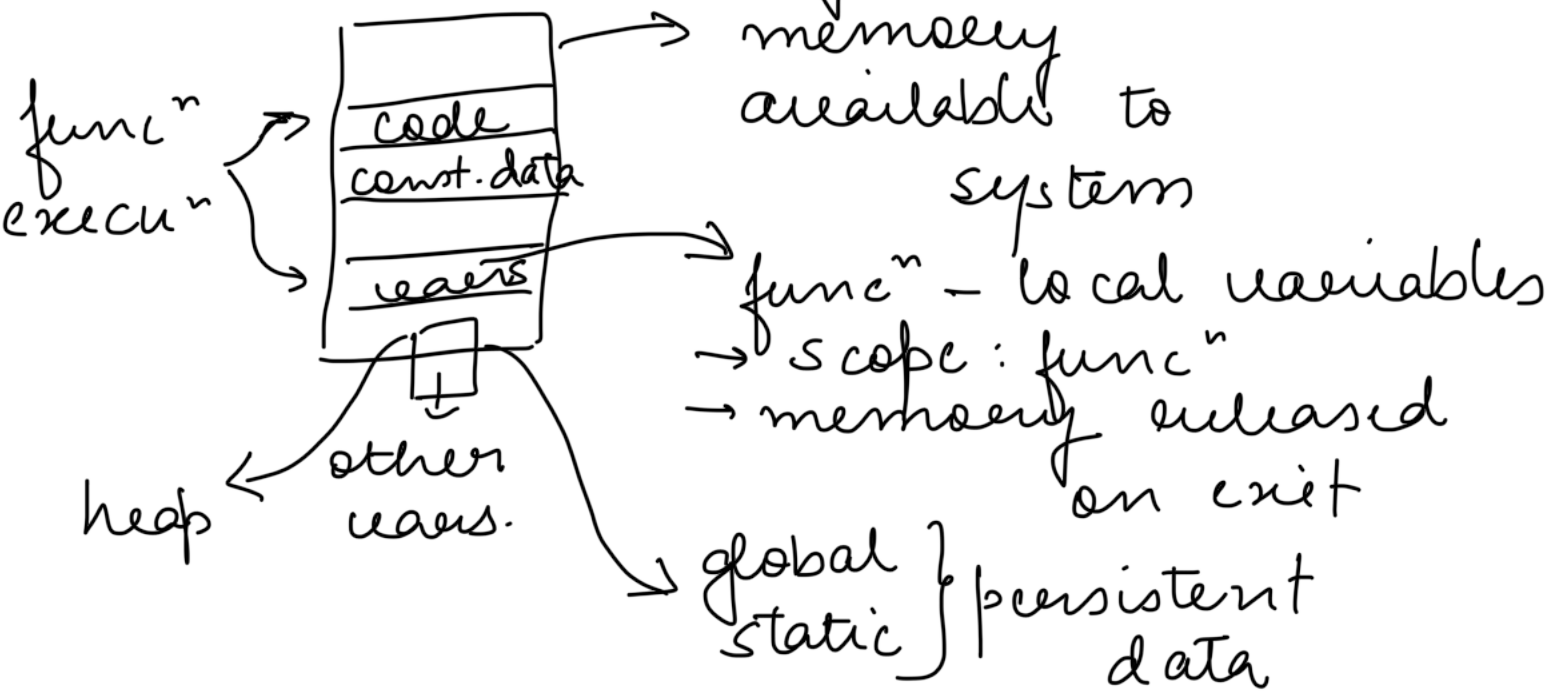
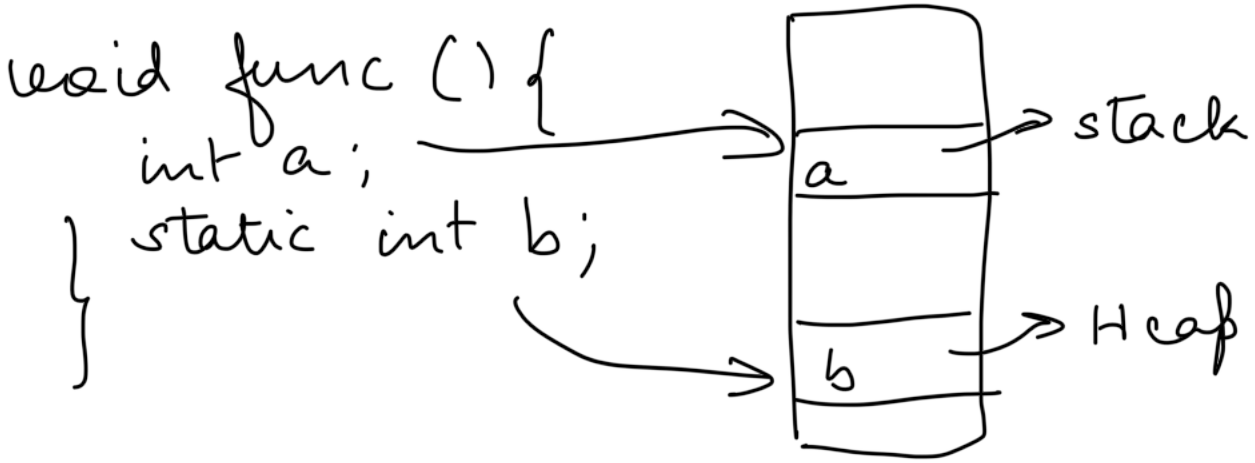
C - Week-9 - Notes

L1 Heap & Stack Frames

- variables & memory





- heap → common across all codes of programs. Static declarations - compile time, global / function static
- Dynamic → manually request / allocate memory. Runtime, explicit use of pointers.

L2. Dynamic Memory Allocⁿ in C

- memory allocⁿ - handled by OS + runtime libs
- request for allocⁿ of memory, specify size of mem block requested
- returns a pointer to chunk of memory or NULL if allocⁿ failed
- once alloted that memory is reserved. Till explicit free or prog exit.
- Possible to access memory w/o reserving
- Very unsafe
- free - explicitly release memory. Available for future allocⁿ. Fragmentaⁿ

- Calloc → aware of type, specify num elements, initialize to 0 (not always desirable)
- realloc → Δ size of allocated memory, may Δ locaⁿ ⇒ memcpy
- arrays don't retain size info., programmer must track.

L3 Egs. of Dynamic Mem. Allocⁿ

```
int *p;
p = (int *) malloc (20 * sizeof (int))
```

L4 Egs. of —— 2

10,000,000

↳ memory created, but when the app. tries to reach that memory it causes segmentation fault (core dumped)

L5 Multi Dim. Array

2D Array us 1D Memory

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \xrightarrow{\text{in mem.}} \left.\begin{bmatrix} a_{11} \\ a_{10} \\ a_{01} \\ a_{00} \end{bmatrix}\right\} \text{flattened}$$

row - major (C)

vs

column-major (fortran)

A[i][j] ⇒ *(A + i*C + j)

int **A ⇒ array of pointers

A[i][j][k] = *(A + i*R*C + j*C + k)