What is the heap?

- The heap is a region of a process's virtual memory space used for dynamic memory allocation at runtime. Heap memory is managed manually by the programmer, i.e. using malloc() and free() in C. Heap memory remains allocated until explicitly freed or until the process terminates.

What is heap alignment and why does the heap need to be aligned?

- Heap alignment refers to the rule that any address returned by the allocator (such as malloc) must be an exact multiple of a fixed power of 2 (typically 8 bytes on 32-bit architectures and 16 bytes on 64-bit architectures.
- Why need alignment?
  ->Modern CPUs access memory across bus lines in word- or double-word-sized chunks. Accessing data that crosses word or cache line boundaries (misaligned access) forces the CPU to issue multiple memory reads and stitch the results together, severely degrading performance.
  ->The C standard mandates that malloc() must return a pointer suitably aligned for any standard data type that can fit into the allocated space. Because the allocator does not know what type of data the caller intends to store, it must default to the strictest fundamental alignment.

What information must the heap store to track allocations?

- Block Size: The total size of the chunk (including headers, padding, and user payload) so the allocator knows how much memory to reclaim upon free().
- Allocation Status: A flag or bit (often packed into the unused low-order bits of the size field due to alignment) indicating whether the block is currently free or allocated.
- Pointers for Free List Navigation: For free blocks, the unused payload space is repurposed to store next and prev pointers to traverse available blocks without scanning the entire heap.
- Previous Block Information (Boundary Tags / Footers): Storing the previous block's size or allocation bit at the boundary allows the allocator to inspect adjacent chunks and perform coalescing.