<u>Compare between stack and heap in RAM.</u>

1. Stack Memory Allocation:
   - Purpose: Stack memory is used for storing local variables within a function.

   - Allocation Mechanism: It happens in contiguous blocks of memory.

   - Lifespan: Variables allocated on the stack are temporary and exist only while the corresponding function is running.

   - Access Control: Data stored in the stack can only be accessed by the owner thread.

   - Memory Management: The compiler automatically allocates and deallocates stack memory for function variables.

   - Safety: Considered safer due to its limited accessibility.

   - Speed: Faster allocation and deallocation compared to heap memory.

   - Storage Space: Stack memory has less storage space compared to heap memory.

2. Heap Memory Allocation:

- Purpose: Heap memory is used for dynamically allocating objects and data structures.
- Allocation Mechanism: It occurs during program execution based on programmer instructions.
- Lifespan: Objects created in the heap have a longer lifespan than stack variables.
- Access Control: Data stored in the heap is accessible to all threads.
- Memory Management: Programmers need to manage heap memory explicitly to avoid memory leaks.
- Categories:
  - Young Generation: Initially allocated for new data (objects), later moved to garbage collection.
  - Old or Tenured Generation: Contains older data objects not frequently used.

In summary, stack memory is suitable for temporary storage, local variables, and function arguments, while heap memory is ideal for large data structures and objects with dynamic lifespans.