ASSIGNMENT -1

In C, **Dynamic Memory Allocation (DMA)** allows programmers to allocate memory during runtime using standard library functions. This is helpful when the size of data structures (like arrays) cannot be determined at compile time.

**Functions of Dynamic Memory Allocation in C:**

1. **malloc() in C – Memory Allocation**

malloc() stands for **memory allocation**. It is a standard library function in C used to **allocate a block of memory dynamically** during program execution.

* Allocates specified number of bytes.
* Returns a pointer to the first byte of the allocated memory.
* Memory is not initialized (contains garbage values).

1. **calloc() in C – Contiguous Allocation**

calloc() stands for **contiguous allocation**. It is used to dynamically allocate memory just like malloc(), but it also **initializes the allocated memory to zero**.

Syntax: calloc(number\_of\_elements, size\_of\_each\_element

1. **realloc() in C – Resize Allocated Memory**

realloc() stands for **re-allocation**. It is used to **resize previously allocated memory** (using malloc(), calloc(), or even earlier realloc()).

* Changes the size of previously allocated memory block.
* Useful to resize arrays when required.

1. **free() in C — Deallocate Memory**

The free() function is used in C to **deallocate memory** that was previously allocated by malloc(), calloc(), or realloc().  
It **frees up heap memory**, making it available for future use.

**Why Dynamic Memory Allocation is useful in C Language?**

**1. Memory Allocation at Runtime**

* You can **allocate memory as needed**, rather than fixing sizes during compile-time.
* Example: Getting array size from the user and allocating memory accordingly.

**2. Efficient Use of Memory**

* Avoids wasting memory by **allocating only what is needed**.
* Releases memory when it's no longer in use (free, delete, garbage collection).

**3. Creating Flexible Data Structures**

* Enables dynamic structures like:
  + **Linked Lists**
  + **Trees**
  + **Graphs**
  + **Stacks & Queues**
* These structures grow/shrink as needed without predefining their size.

**4. Handling Large Data**

* Static allocation is limited to the stack, which has size limits.
* Dynamic memory uses the **heap**, which is much larger and better for large datasets.

**5. Multiple Instances**

* You can create many instances of objects or arrays at runtime using dynamic memory.