Summary_Task6

1. Pointers and Arrays in Structures

- A structure in C can include arrays (like char name [30]) and also be used with pointers.
- Arrays in a structure hold multiple values of the same type (e.g., strings as arrays of characters).
- You can also use pointers to structures to access or modify structure members using the -> operator.
  - Example: ptr->name is the same as (*ptr).name
- This is useful for passing structures to functions or working with dynamic memory.

2. Passing a Structure to a Function

There are two common ways:

- **By Value** – A *copy* of the structure is sent to the function.
  - Changes inside the function do **not** affect the original.
- **By Pointer** – A pointer to the structure is passed.
  - Changes inside the function **do affect** the original structure.
  - This is memory-efficient and preferred for large structs.

3. Size of a Structure

- The **size** of a structure is calculated using sizeof(struct myStruct).
- It includes the total bytes used by all members, **plus any extra bytes added for alignment** (see padding below

4. Memory Padding and Alignment

What is Padding?

- **Padding** is extra unused memory the compiler adds between structure members.
- It's added to make sure each variable starts at a memory address that’s efficient for the CPU to access.

Aligned Memory

- Data is stored at memory addresses that match the CPU’s natural access size (like 4 bytes for int).
- Fast access, no CPU penalty.

Unaligned Memory

- If structure members are not properly aligned (for example using #pragma pack(1)), access can be **slower** or may even cause **hardware errors** on some systems.

Concept	Structure (C)	Object
Data only	Holds data fields	Holds both data **and** methods
No encapsulation	Data is public by default	Supports **encapsulation** (private/public)
No inheritance	Cannot inherit from another struct	**Supports inheritance** (OOP feature)
Procedural approach	Used in procedural programming	Used in **object-oriented** programming
Lightweight	Simple and low-overhead	More powerful but adds complexity