Summary

Structures: Is a user defined datatype that can be used to group elements of different types into a single type.

Ex:

```
struct employee{
    char name[20];
    int age;
    int salary;
};
```

To initialize structure variables -> struct employee emp = {“Mariam”, 30, 5000};

To access structure variables -> using dot operator -> emp.name = “Mariam”;

Arrays with structures:

```
#include <stdio.h>
struct car {
    int speed;
    int fuel;
};
int main() {
    struct car c[2];
    for (int i = 0; i < 2; i++) {
        printf("Enter speed of car %d: ", i + 1);
        scanf("%d", &c[i].speed);
        printf("Enter fuel of car %d: ", i + 1);
        scanf("%d", &c[i].fuel);
    }
    for (int i = 0; i < 2; i++) {
        printf("Car %d\n - Speed: %d, Fuel: %d\n", i+1, c[i].speed, c[i].fuel);
    }
    return 0;
}
```

Pointers with structures:

```c
#include <stdio.h>
struct Date {
    int day;
    int month;
    int year;
};

int main() {
    struct Date date1, date2;
    struct Date *p = &date1;
    struct Date *q = &date2;

    if(p->year == q->year
       && p->month == q->month
        && p->day == q->day)
        printf("Dates are equal\n");
        else
        printf("Dates are not equal\n");

    return 0;
}
```

Size of a structure:

The size of a structure = sizes of its members + padding added.

To get the size we use -> sizeof(struct_name) as shown in problem 4

Memory padding: Extra unused bytes added by the compiler between structure members to align data in memory

memory	**Aligned**	**Unaligned**
speed	faster	slower
size	larger	smaller

Difference between the Structures and Object

	Structues	Object (in OOP)
Inhertance	Not supported	Supports it
Access control	No access control (no private/ public)	Supports encapsulation with access specifiers
behavior	Can't act	Can have methods / functions built in

Conclusion: Both are containers while objects can have actions.