Chapter 10 – Structures and Unions

✅ Short Questions with Easy Answers

Q1. Define Structure.

👉 A **structure** is a user-defined datatype that allows grouping of **different data types** under one name.

- Example:

```
struct Student {
  int rollNo;
  char name[20];
  float marks;
};
```

Q2. Define Union.

👉 A **union** is similar to a structure, but **all members share the same memory location**. Memory size = largest member.

- Example:

```
union Data {
  int id;
  float salary;
  char grade;
};
```

Q3. How many data types can structure fields use?

👉 Structure fields can use **all C data types** like int, float, char, double.

Q4. Benefits of Structures?

- Groups related data together.
- Makes code **organized and easier to maintain**.
- Useful for keeping records (like student data).
- Whole record can be accessed using one variable.

Q5. How to declare a structure variable?

👉

```
struct Student {
  int rollNo;
  char name[20];
} s1;   // structure variable s1
```

Q6. How to access Union members?

👉 Use . (dot) operator or -> (arrow with pointer).

```
union Data d;

d.id = 10;

d.salary = 5000.50;
```

Q7. Define array of structures.

👉 Multiple structure variables in one array.

```
struct Student {
  int id;
  float marks;
};

struct Student st[5];  // array of 5 students
```

Q8. Syntax of a Union.

👉

```
union UnionName {
  data_type member1;
  data_type member2;
};
```

Q9. Define union of structures.

👉 A union can contain structures, and a structure can contain unions.

```
union Example {
  struct {
    int x, y;
  } point;
  float z;
};
```

Q10. What is the use of typedef in structures?

👉 Makes code **short and readable**.

```
typedef struct {
  int rollNo;
  char name[20];
} Student;


Student s1;  // no need to write struct keyword
```

✅ Long Questions

Q1. How to create a Structure in C? (with example)

👉 Example:

```
#include <stdio.h>
struct Student {
  int rollNo;
  char name[20];
  float marks;
};
int main() {
  struct Student s1 = {1, "Ali", 88.5};
  printf("Roll No: %d\n", s1.rollNo);
  printf("Name: %s\n", s1.name);
  printf("Marks: %.2f\n", s1.marks);
  return 0;
}
```

Q2. What are Unions? How to define, declare, and initialize?

👉 Example:

```
#include <stdio.h>
union Data {
  int id;
  float salary;
  char grade;
};
int main() {
  union Data d1;
  d1.id = 101;   // uses memory for int
  printf("ID = %d\n", d1.id);
  d1.salary = 2500.50; // overwrites same memory
  printf("Salary = %.2f\n", d1.salary);
  return 0;
}
```

⚠ Only **one member is valid at a time** in a union.

Q3. Differentiate between Structure and Pointer (with example).

- A **structure** groups different types of variables.
- A **pointer** stores the address of a variable.
- A **pointer to structure** is used to access structure members with ->.

👉 Example:

```
struct Student {
  int rollNo;
  char name[20];
};
int main() {
  struct Student s1 = {101, "Ali"};
  struct Student *ptr = &s1;
  printf("%d %s", ptr->rollNo, ptr->name);
  return 0;
}
```

Q4. Difference between Structure and Union.

Feature	Structure	Union
Memory usage	Each member has separate memory	All members share same memory
Size	Sum of all members	Size of largest member
Access	All members can be used at the same time	Only one member is valid at a time
Example use	Student record (name, roll, marks)	Store multiple representations of data

---

Q5. Write a note on Union of Structures.

👉 Just like a union can hold different data types, it can also hold **structures inside** it. This is useful when only one of several structure types is needed at a time.

🔷 **Extra Practice Questions**

Q1. Write a program to input and display 3 students' records using array of structures.

```
#include <stdio.h>
struct Student {
  int rollNo;
  char name[20];
```

```
  float marks;
};
int main() {
  struct Student s[3];
  for(int i=0;i<3;i++) {
    printf("Enter roll, name, marks: ");
    scanf("%d %s %f", &s[i].rollNo, s[i].name, &s[i].marks);
  }
  printf("\nStudent Records:\n");
  for(int i=0;i<3;i++) {
    printf("%d %s %.2f\n", s[i].rollNo, s[i].name, s[i].marks);
  }
  return 0;
}
```

Q2. What is difference between typedef struct and normal struct?

- struct Student s1; (normal way)
- typedef struct { ... } Student; Student s1; (short way)

Q3. Program using Union.

```
#include <stdio.h>
union Item {
  int id;
  float price;
};
int main() {
  union Item i1;
  i1.id = 10;
  printf("ID: %d\n", i1.id);
  i1.price = 99.5;
  printf("Price: %.2f\n", i1.price);
  return 0;
}
```