1. Theory Questions (with some tricky parts)

1. **Explain the difference between structures and unions in C.** When would you prefer a union over a structure?

2. **What is a bit field, and how does it help with memory efficiency?** Are there any limitations to using bit fields in C?

3. **Discuss the alignment and padding in structures.** How can you minimize padding, and how does it impact performance?

4. **Why are bitwise operations often used in low-level programming?** Provide an example where using bitwise operations is more efficient than other alternatives.

5. **Can you use an array within a structure?** How would you dynamically allocate memory for such an array?

6. **Explain how unions work with floating-point and integer values.** What would happen if you assign an integer to a union and try to retrieve it as a float?

7. **What are the potential issues with using a pointer to a structure?** Explain how pointers affect memory access within structures.

8. **What are some common pitfalls with using bitwise operators?** Explain with examples.

9. **Describe the memory layout of a structure with bit fields.** Why might the order of bit fields affect the overall memory footprint?

10. **How does using typedef with structures improve code readability and maintainability?** Provide an example of a typedef structure with nested elements.

---

2. Output Type Questions

1. What is the output? **Consider byte-ordering (endianness) effects**:

```c
C/C++
union {
    int num;
    char bytes[4];
} u;
u.num = 0x12345678;
printf("%x %x %x %x", u.bytes[0], u.bytes[1], u.bytes[2], u.bytes[3]);
```

2. Predict the output of this code:

```C/C++
struct data {
    unsigned int a : 3;
    unsigned int b : 5;
} d = {7, 31};
printf("%d %d", d.a, d.b);
d.a += 1; d.b += 1;
printf("%d %d", d.a, d.b);
```

3. Given this union, what is the output?

```C/C++
union {
    int x;
    char c[4];
} u = { .x = 256 };
printf("%d", u.c[1]);
```

4. Consider this structure. What does it print, and why?

```C/C++
struct {
    int x : 3;
    int y : 3;
} s = {5, -3};
printf("%d %d", s.x, s.y);
```

5. Analyze the output:

```c
C/C++
struct person {
    int id;
    char name[4];
} p1 = {1, "John"};
printf("%d %s", p1.id, p1.name);
```

6. **Trick question**: What is the output here, and why?

```c
C/C++
struct {
    int x;
    char c;
    double d;
} s;
printf("%zu", sizeof(s));
```

7. Predict the result:

```c
C/C++
struct point {
    int x : 2;
    int y : 4;
} p = {3, 9};
printf("%d %d", p.x, p.y);
```

8. Given the code below, what is the output?

```c
C/C++
struct {
    int a : 4;
    unsigned int b : 4;
} s = {10, 15};
printf("%d %u", s.a, s.b);
```

9. What is the result here?

```
C/C++
struct {
    unsigned a : 1;
    unsigned b : 1;
} s = {1, 1};
s.a ^= s.b;
printf("%u %u", s.a, s.b);
```

10. **Challenge**: Analyze this bit manipulation output:

```
C/C++
int x = 10;
x = x & ~(1 << 1);
printf("%d", x);
```

---

3. Programming Questions

1. Define a structure with nested bit fields to hold RGB color values (8 bits each for R, G, and B). **Write a function to extract and print each color component from a single 24-bit integer.**

2. **Write a program to find if a given integer is a power of 2 using bitwise operations only.**

3. Create a structure to store a date (day, month, year). **Write a function that takes two dates and calculates the number of days between them. Handle leap years.**

4. **Define a union to store either a character array or an integer array of size 4.** Write a function to alternate between interpreting the union as a string or integer array based on user input.

5. Define a structure with a bit field that restricts an integer to 5 bits. **Write a program that attempts to store numbers larger than 31 and explain what happens.**

6. **Write a program that uses bitwise operations to set, clear, and toggle bits of an integer.** Provide a menu for each operation.

7. **Using structures and arrays of structures, create a simple student grading system** where each student has an array of grades. Write functions to add grades and calculate each student's average.

8. Write a program that reads a union containing both an `int` and a `float`. **Observe and print the changes in both fields when values are assigned to one of them.**

9. Define a structure for a 3D point with bit fields to limit `x`, `y`, and `z` to a range of -8 to 7. **Write a function to add and print the coordinates of two 3D points.**

10. **Implement a program that shifts the bits of an integer to the left or right by a given number of positions.** Prompt the user for the direction and number of positions.

11. Write a program to accept an array of structures, each holding an integer and a character. **Sort the array based on integer values and print the result.**

12. **Implement a program to reverse the bits of an integer.** Print both the original and reversed values.

13. Write a program to store and display complex numbers using structures. **Add functions for addition, subtraction, and multiplication of complex numbers.**

14. **Create a program that finds the position of the first 1-bit in a given integer using bitwise operations.** Output the position from the right (least significant bit).

15. **Use a union to interpret a floating-point number as an integer.** Print the binary representation of a float by accessing it through the integer part of the union.