Embedded C Practice Questions

1. Bit Manipulation & Register Control

- Write a macro to set, clear, toggle, and read a bit at a specific position.
- Check if a given number is a power of 2.
- Write a function to count set bits in a byte.
- Swap the nibbles in an 8-bit number.
- Reverse bits in an 8-bit or 32-bit number.
- Implement BIT(n) macro that returns (1 << n).

2. Pointers, Arrays & Memory

- Write your own version of memcpy, memset, and memcmp.
- Reverse an array using only pointer arithmetic.
- Write a function to simulate malloc() from a static array (fixed-size memory pool).
- Implement pointer-to-pointer usage to dynamically allocate a 2D array.
- Explain the difference between pointer to constant and constant pointer (and use them in code).

3. Circular Buffer & Data Queues

- Implement a circular buffer (ring buffer) with push() and pop() APIs.
- Extend it to be used for UART RX/TX buffer simulation.
- Add overflow/underflow checks to the circular buffer.

4. Embedded System Patterns

- Implement a finite state machine (FSM) for: Traffic light controller, Button debounce, UART command parser.
- Write a main event loop (while(1)) that calls different tasks at different intervals.
- Simulate a scheduler (round-robin or cooperative multitasking).

5. Structs, Enums & Configuration

- Define a struct for a sensor device with fields like ID, state, value.

- Create an enum to represent device states, and write a function to change state.

- Implement a configuration structure with compile-time default values using macros.

6. Communication Protocol Simulation

- Implement CRC8 or CRC16 checksum generation.

- Write a function to encode/decode using COBS (Consistent Overhead Byte Stuffing).

- Parse a hex string command like 0xAA 0x01 0x03 0x04 and extract header/data fields.

- Create a function that simulates a command-response protocol over UART.

7. RTOS-Like Task Simulation

- Implement task_create, task_delay, task_run in bare-metal style C.

- Schedule 3 tasks that run at different periodic intervals (e.g., 10ms, 50ms, 100ms).

- Simulate a priority-based task execution model.

8. File I/O & Testing (on Linux PC)

- Read/write a binary file using fread, fwrite to simulate EEPROM.

- Log sensor data to a file, then parse and compute average/min/max.

- Write unit tests in C using basic assert() macros for your buffer or protocol code.

9. Miscellaneous Embedded Concepts

- Convert blocking delay (delay_ms) to non-blocking delay using timestamp comparison.

- Write a mock driver for a GPIO or LED (e.g., LED_On(), LED_Off()).

- Create a mock I2C transaction between master and slave devices.

- Simulate a watchdog timer that resets if a function is not called within n ms.

10. Debugging & Optimization

- Identify and fix memory leak in a program using valgrind.

- Optimize a C function for memory usage - e.g., removing dynamic allocation.

- Write macros for debug printing with file, line, and function name info.