Top 100 Embedded C Programming Interview Questions

1. C Language Core (Syntax & Logic)

• Reverse a string without using library functions.
• Implement strcmp(), strcpy(), and strlen() manually.
• Count number of set bits in an integer.
• Swap two numbers without using a temporary variable.
• Check if a number is a power of two.
• Reverse bits of an 8-bit or 32-bit number.
• Find the endianness of your system (big or little).
• Implement your own version of atoi().
• Implement a simple state machine in C.
• Print Fibonacci sequence without recursion.

2. Bitwise Operations

• Toggle the nth bit of a number.
• Count leading/trailing zeros in an integer.
• Extract bits from position p to q.
• Merge two 8-bit values into one 16-bit variable.
• Pack/unpack multiple sensor data fields into uint32_t.
• Implement circular bit rotation.
• Check if two numbers have opposite signs.
• Write macros to set, clear, toggle, check bits.
• Implement software debounce using bits.
• Convert 12-bit ADC raw data to voltage.

3. Pointers and Memory

• Difference between pointer to constant and constant pointer.
• Implement a function pointer and call dynamically.
• Return a pointer to static variable.
• Use pointer to structure and access members.
• Implement singly/doubly linked list.
• Flatten a doubly linked list with child pointers.
• Detect and remove loops in linked lists.
• Implement stack and queue using pointers.
• Reverse a linked list.
• Deep vs shallow copy example.

4. Arrays and Strings

• Rotate an array by k positions.
• Find duplicates in an array.
• Find the missing number in a sequence.
• Find maximum and minimum in an array.
• Implement matrix multiplication.
• Check if two strings are anagrams.
• Implement substring search (naive or KMP).
• Convert uppercase to lowercase manually.
• Implement circular buffer.
• Implement ring buffer overflow handling.

5. Control Flow & Algorithms

• Implement binary search.
• Implement bubble/quick/insertion sort.
• Find GCD and LCM of two numbers.
• Implement integer square root function.
• Find factorial using recursion and iteration.
• Implement moving average filter.
• Implement debouncing logic.
• Implement checksum (8/16-bit).
• Implement CRC-8 or CRC-16.
• Simulate PWM generation.

6. Embedded-Specific Concepts

• Implement GPIO control macros.
• Simulate UART communication TX/RX buffer.
• Write ISR-like handler logic.
• Implement watchdog timer reset simulation.
• Detect system reset cause (POR/BOD/WDT).
• Implement FSM for LED blinking.
• Simulate ADC read and averaging.
• Implement SPI transmit/receive buffer.
• Create digital low-pass filter.
• Simulate temperature sensor with threshold.

7. Hardware Abstraction and Drivers

• Driver initialization for UART/SPI.
• GPIO read/write APIs.
• Delay using hardware timer.
• Round robin scheduler.
• I2C bit-banging implementation.
• LCD driver using SPI/parallel.

• ADC-to-temperature conversion.
• EEPROM store/retrieve data.
• Non-blocking UART transmit.
• LEDs and switches driver.

8. Real-Time and System Level

• Cooperative multitasking system.
• Task scheduler using timer interrupts.
• Simulate RTOS task switching.
• Producer-consumer queue.
• Semaphore-like mechanism in C.
• Demonstrate priority inversion fix.
• Time slicing between tasks.
• Soft real-time timer handling.
• Event-driven system simulation.
• Message passing between tasks.

9. Memory Management

• Simple malloc/free simulation.
• Stack overflow detection.
• Simulate data, BSS, and stack segments.
• Variable placement in memory sections.
• Static vs dynamic allocation.
• DMA circular buffer handling.
• Share memory between ISR and main.
• Implement memcpy manually.
• Demonstrate volatile usage.
• Structure padding/alignment example.

10. Miscellaneous / Interview-Style

• main() before hardware initialization impact.
• Interrupt latency and ISR safety.
• Handle multiple tasks with limited stack.
• Why volatile is important (example).
• Safe circular queue for UART.
• Error codes and handling mechanism.
• Static vs global vs extern variable example.
• Temperature control logic (heater ON/OFF).
• Power sequencing logic for multiple rails.
• Watchdog-triggered reset and recovery.

Bonus: Practical Embedded Challenges

- Interface a temperature sensor and display value on LCD.
- Simulate UART command parsing and response generation.
- Implement system power-on self-test (POST).
- Log data into EEPROM with timestamp rollover.
- Design a bootloader-like firmware updater.