100 C Programming Questions for Embedded Firmware Development

Part 1: Basics of C

Write a program to swap two numbers without using a third variable.

Find factorial of a number using recursion and iteration.

Check if a number is prime.

Reverse a number.

Check if a string is a palindrome.

Count vowels and consonants in a string.

Convert lowercase to uppercase without using toupper().

Implement your own version of strlen().

Implement a basic calculator using switch.

Print Fibonacci series using recursion.

Part 2: Arrays and Strings

Reverse an array.

Find the largest and smallest number in an array.

Sort an array using bubble sort.

Remove duplicates from an array.

Implement linear search.

Implement binary search.

Count frequency of each character in a string.

Find the first non-repeating character in a string.

Check if two strings are anagrams.

Implement your own version of strcpy().

Part 3: Pointers and Memory

Demonstrate pointer arithmetic with arrays.

Write a program to reverse a string using pointers.

Implement malloc() and free() usage.

Pass pointer to function and modify value.

Implement pointer to pointer operations.

Create a dynamic array using malloc.

Compare two strings using pointers.

Implement your own version of strcmp().

Demonstrate usage of const with pointers.

Difference between char *str and char str[].

Write a function that takes an array and size, and reverses it.

Allocate memory for 2D array dynamically.

Implement memory leak example and fix it.

Explain and demonstrate dangling pointer.

Demonstrate wild pointer and how to avoid it.

Part 4: Functions and Recursion

Write a function to find GCD using recursion.

Tail vs Head recursion - write examples.

Implement a recursive binary search.

Implement recursive reverse of a string.

Implement recursive factorial and Fibonacci.

Count number of digits using recursion.

Write pow(x, n) using recursion.

Implement a recursive palindrome check.

Demonstrate static vs non-static functions.

Implement function pointer usage in a menu-driven program.

Part 5: Structures and Unions

Define a structure for a student and print details.

Implement a structure with nested structure.

Compare memory usage of structure and union.

Implement bit fields in structure.

Pass structure to a function by value and reference.

Implement an array of structures.

Store and retrieve data of students using file I/O.

Use typedef with structures.

Create a structure with pointer members and allocate memory.

Serialize and deserialize structure data.

Part 6: Bit Manipulation

Count number of 1s in a binary number.

Check if a number is power of 2.

Reverse bits of a number.

Set, clear, toggle, and check a bit at given position.

Write macros for bit operations.

Swap two numbers using XOR.

Find the only non-repeating element in an array using XOR.

Implement a circular bit shift.

Find the position of the first set bit.

Use bit-fields to manage flags efficiently.

Part 7: Storage Classes and Qualifiers

Demonstrate static variable in a function.

Use register storage class and observe behavior.

Use volatile with global variables and simulate interrupt behavior.

Use const with pointers and functions.

Explain scope and lifetime of variables using code.

Part 8: Preprocessor and Macros

Write a macro to get max of two numbers.

Write a macro that squares a number properly.

Use #ifdef, #ifndef for conditional compilation.

Implement token pasting and stringizing macros.

Simulate a logging macro that logs function name and line number.

Part 9: Embedded C Specific and Hardware Simulated Questions

Blink an LED using a delay loop (pseudocode).

Implement a delay function using timers (assume 8-bit timer).

Create a function to set a bit in a register.

Simulate register map using structures.

Write ISR function prototype (as used in embedded).

Explain how to debounce a button in code.

Write a simple state machine in C.

Simulate UART data transmit buffer.

Implement a circular buffer.

Simulate GPIO read/write using macros.

Write a function to map a pin number to port and bit mask.

Define a structure to simulate memory-mapped peripheral.

Implement simple CRC check algorithm.

Write a watchdog timer reset simulation.

Implement fixed-point arithmetic function.

Part 10: Interview-style & Advanced Questions

Explain stack vs heap with code example.

Implement a custom malloc (simplified).

Detect memory corruption using canary method.

Implement memcpy function.

Implement strtok function.

Simulate a simple scheduler with function pointers.

Implement a timeout mechanism using a timer counter.

Write a test case for circular buffer.

Write code to toggle a GPIO at a given interval.

Optimize a function for memory usage.