```c
/*
Question 1:
Write a C program that swaps two integer variables without using a third variable or arithmetic
operators (+, -, *, /).
Use bitwise operators to achieve the swap.

Author: Meher Chaitanya
Date: 24/09/25

Version 1 - The Black Magic Method
*/

#include<stdio.h>

void main() {
    // Warning: Works only with integers < 65536 (half int size)
    int a = 7;
    int b = 10;

    printf("%d %d\n", a, b); // OUT: 7 10

    // left-shift `a` by half of int size (4*8) to make space for `b` insertion
    a = a << 2*8;
    // insert `b` into right-end of `a`
    a = a | b;
    // right-shift (undo) `a` to get back original `a` and assign to `b`
    b = a >> 2*8;
    // perform black magic to clear left-half of `a` to keep only value of `b`
    a = a & ~(2*8-1 << 2*8);

    printf("%d %d\n", a, b); // OUT: 10 7
}
```

```c
/*
Question 1:
Write a C program that swaps two integer variables without using a third variable or arithmetic
operators (+, -, *, /).
Use bitwise operators to achieve the swap.

Author: Meher Chaitanya
Date: 24/09/25

Version 2 - Recursive Function Method
*/

#include<stdio.h>

int add(int a, int b) {
    // Recursive implementation of bitwise addition
    if (b == 0) {
        return a;
    }
    // Bitwise implementation of logic gate equivalent of addition
    return add(a^b, (a&b)<<1);
}

int sub(int a, int b) {
    // Recursive implementation of bitwise subtraction
    if (b == 0){
        return a;
    }
    // Bitwise implementation of logic gate equivalent of subtraction
    return sub(a^b, (~a&b)<<1);
}

void main() {
    int a = 7;
    int b = 10;

    printf("%d %d\n", a, b); // OUT: 7 10

    /*
    The following is a bitwise emulation of:
        a = a + b
        b = a - b
        a = a - b
    */

    a = add(a, b);
    b = sub(a, b);
    a = sub(a, b);

    printf("%d %d\n", a, b); // OUT: 10 7
}
```

```c
/*
Question 2:
Write a function in C that takes an integer n and returns:
  * "Prime" if n is a prime number,
  * "Perfect Square" if n is a perfect square,
  * "Neither" otherwise.
Your program should handle edge cases (e.g., n <= 1) correctly.

Author: Meher Chaitanya
Date: 24/09/25
*/

#include<stdio.h>
#include<stdlib.h>

char * analyze_number(int n) {
    int is_prime = 1; // prime if no factor found
    for (int i=2; i<=n/2; i++) {
        if (n%i==0) {
            is_prime = 0; // factor found, composite
        }
    }

    if (is_prime) {
        return "Prime"; // return if prime, cannot be perfect square
    }

    int is_perfect_square = 0; // not perfect square by default
    for (int i=2; i<=n/2; i++) {
        if (i*i==n) {
            is_perfect_square = 1; // perfect square if i^2 == n
            break;
        }
    }

    if (is_perfect_square) {
        return "Perfect Square"; // return if perfect square
    } else {
        return "Neither"; // return if neither prime nor perfect square
    }
}

void main() {
    int n;
    printf("Enter an number: ");
    scanf("%d", &n); // take int input

    if (n <= 1) { // guard clause to check for edge case
        printf("Please enter a number greater than 1.\n");
        exit(1); // exit with error
    }

    printf("Result: %s\n", analyze_number(n)); // run function and print output
}
```

```c
/*
Question 3:
Write a C program to print the first n numbers in the Fibonacci sequence
using a for loop, but without using any additional variables except for
the current and previous two numbers.

Author: Meher Chaitanya
Date: 24/09/25
*/

#include<stdio.h>
#include<stdlib.h>

void main() {
    int n;
    printf("Enter number of Fibonacci numbers to generate: ");
    scanf("%d", &n); // take int input

    if (n < 1) { // guard clause to check if input not natural number
        printf("Please enter a count greater than 0.\n");
        exit(1); // exit with error
    }

    long p=0, q=1, c=p;
    for (int i=0; i<n; i++) { // iterate n times
        printf("%ld\n", c);
        c = p+q;
        p = q;
        q = c;
    }
}
```

```c
/*
Question 4:
Write a C function that takes an array of integers and its size, then
returns the length of the longest contiguous subarray where the difference
between consecutive elements is exactly 1 or -1.

Example:
Input: [2, 3, 4, 3, 2, 1, 2]
Output: 5 (for the subarray [4, 3, 2, 1, 2])

Author: Meher Chaitanya
Date: 24/09/25
*/

#include<stdio.h>
#include<stdlib.h>

void main() {
    int n;
    printf("Enter size of array you would like to input: ");
    scanf("%d", &n);

    if (n < 1) { // guard clause
        printf("Please choose a size greater than 0.\n");
        exit(1);
    }

    int arr[n];
    printf("Enter %d value(s):\n", n);
    for (int i=0; i<n; i++) { // take n inputs
        scanf("%d", &arr[i]);
    }

    int l = 1; // length of longest contiguous subarray = 1 (always minimum)
    for (int i=1; i<n; i++) {
        if (abs(arr[i]-arr[i-1]) == 1) { // check if difference is 1 or -1
            l += 1; // increment by 1
        } else {
            l = 1; // reset to 1
        }
    }
    printf("Length of longest contiguous subarray: %d\n", l);
}
```

```c
/*
Question 5:
Write a C program that takes an array of integers and separates the numbers into two arrays:
one containing all even numbers, and the other containing all odd numbers.
Then, print the two arrays in reverse order using only pointer arithmetic (no indexing).

Author: Meher Chaitanya
Date: 24/09/25
*/

#include<stdio.h>
#include<stdlib.h>

void main() {
    int n;
    printf("Enter the size of array you would like to input: ");
    scanf("%d", &n);

    if (n < 1) { // guard clause
        printf("Please choose a size greater than 0.\n");
        exit(1);
    }

    int arr[n];
    printf("Enter %d value(s):\n", n);
    for (int i=0; i<n; i++) { // take n inputs
        scanf("%d", &arr[i]);
    }

    int even_arr[n], odd_arr[n];
    int e_i=0, o_i=0; // hold indices for even_arr and odd_arr
    for (int i=0; i<n; i++) {
        if (arr[i]%2==0) { // check if even
            even_arr[e_i] = arr[i]; // append to even_arr
            e_i++;
        } else {
            odd_arr[o_i] = arr[i]; // append to odd_arr
            o_i++;
        }
    }

    printf("Even numbers: ");
    int *e_pt = even_arr; // get pointer of first element in even_arr
    for (int i=0; i<e_i; i++) {
        printf("%d ", *(e_pt+i)); // get value of any element in even_arr using pointer
    }
    printf("\n");

    printf("Odd numbers: ");
    int *o_pt = odd_arr; // get pointer of first element in odd_arr
    for (int i=0; i<o_i; i++) {
        printf("%d ", *(o_pt+i)); // get value of any element in odd_arr using pointer
    }
    printf("\n");
}
```