```c

// Function definitions
int add(int a, int b) {
    return a + b;
}

int subtract(int a, int b) {
    return a - b;
}

int multiply(int a, int b) {
    return a * b;
}

float divide(int a, int b) {
    return (float)a / b;  // Casting to float for decimal result
}

```

```c
#include <stdio.h>
#include "Q1_Calculator.h"

int main() {
    int num1, num2, choice;

    do {
        printf("Enter two integers: ");
        scanf("%d %d", &num1, &num2);

        printf("\nChoose an operation:\n");
        printf("1. Add\n2. Subtract\n3. Multiply\n4. Divide\n5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            printf("Sum = %d\n", add(num1, num2));
            break;
        case 2:
            printf("Difference = %d\n", subtract(num1, num2));
            break;
        case 3:
            printf("Product = %d\n", multiply(num1, num2));
            break;
        case 4:
            if (num2 != 0)
                printf("Quotient = %.2f\n", divide(num1, num2));
            else
                printf("Error: Division by zero!\n");
            break;
        case 5:
            printf("Exiting the program.\n");
            break;
        default:
            printf("Invalid choice! Please try again.\n");
        }

        printf("\n");

    } while (choice != 5);  // Exit condition when choice is 5

    return 0;
}

```

```c
#include <stdio.h>

// Recursive function to calculate the sum of natural numbers
int sum_of_natural_numbers(int n) {
    if (n == 1)  // Base case: if n is 1, return 1
        return 1;
    else
        return n + sum_of_natural_numbers(n - 1);  // Recursive call
}

int main() {
    int n = 20;
    int result = sum_of_natural_numbers(n);
    printf("The sum of the first 20 natural numbers is: %d\n", result);
    return 0;
}

```

```c
#include <stdio.h>

// Recursive function to calculate Fibonacci numbers
int fibonacci(int n) {
    if (n == 0)  // Base case 1: Fibonacci(0) = 0
        return 0;
    else if (n == 1)  // Base case 2: Fibonacci(1) = 1
        return 1;
    else
        return fibonacci(n - 1) + fibonacci(n - 2);  // Recursive case
}

int main() {
    int n, i;

    printf("Enter the number of terms in the Fibonacci series: ");
    scanf("%d", &n);

    printf("Fibonacci series:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i));
    }
    printf("\n");

    return 0;
}

```

```c
#include <stdio.h>

// Function for swapping using call-by-value
void swap_by_value(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
    printf("Inside swap_by_value: a = %d, b = %d\n", a, b);
}

// Function for swapping using call-by-reference
void swap_by_reference(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
    printf("Inside swap_by_value: a = %d, b = %d\n", *a, *b);
}

int main() {
    int num1, num2;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    // Swapping using call-by-value
    printf("\nBefore swap_by_value: num1 = %d, num2 = %d\n", num1, num2);
    swap_by_value(num1, num2);
    printf("After swap_by_value: num1 = %d, num2 = %d\n", num1, num2);  // No change in main

    // Swapping using call-by-reference
    printf("\nBefore swap_by_reference: num1 = %d, num2 = %d\n", num1, num2);
    swap_by_reference(&num1, &num2);
    printf("After swap_by_reference: num1 = %d, num2 = %d\n", num1, num2);  // Changes reflected
 in main

    return 0;
}

```

```c
#include <stdio.h>

// Recursive function to calculate the sum of digits
int sum_of_digits(int n) {
    if (n == 0)  // Base case: if the number is 0, return 0
        return 0;
    else
        return (n % 10) + sum_of_digits(n / 10);  // Recursive case
}

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    // Handling negative numbers by converting them to positive
    if (num < 0) {
        num = -num;
    }

    int result = sum_of_digits(num);
    printf("The sum of the digits is: %d\n", result);

    return 0;
}

```

```c
#include <stdio.h>

// Recursive function to reverse the digits of the number
void reverse_number(int n) {
    if (n == 0)
        return;  // Base case: if the number is 0, stop recursion
    else {
        printf("%d", n % 10);  // Print the last digit
        reverse_number(n / 10);  // Recursive call with the remaining digits
    }
}

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    // Handling negative numbers
    if (num < 0) {
        printf("-");
        num = -num;  // Convert negative number to positive
    }

    printf("The reverse of the number is: ");

    if (num == 0)
        printf("0");  // Special case for 0

    reverse_number(num);  // Call the recursive function
    printf("\n");

    return 0;
}

```

```c
// Function definition for finding maximum
int find_max(int a, int b) {
    if (a > b)
        return a;
    else
        return b;
}

// Function definition for finding minimum
int find_min(int a, int b) {
    if (a < b)
        return a;
    else
        return b;
}
```

```c
#include<stdio.h>
#include "Q7_Max_Min.h"
int main() {
    int num1, num2, max, min;

    // Input two numbers from the user
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    // Find the maximum and minimum using the functions
    max = find_max(num1, num2);
    min = find_min(num1, num2);

    // Output the results
    printf("Maximum: %d\n", max);
    printf("Minimum: %d\n", min);

    return 0;
}
```

```c
#include <stdio.h>

// Function declaration to check even or odd
const char* check_even_odd(int num);

int main() {
    int number;

    // Input a number from the user
    printf("Enter an integer: ");
    scanf("%d", &number);

    // Call the function and print the result
    printf("The number %d is %s.\n", number, check_even_odd(number));

    return 0;
}

// Function definition to check even or odd
const char* check_even_odd(int num) {
    if (num % 2 == 0) {
        return "even";  // Number is even
    } else {
        return "odd";   // Number is odd
    }
}

```

```c
    return 1; // Prime number
}

// Function to check if a number is an Armstrong number
int is_armstrong(int num) {
    int sum = 0, original_num = num;
    int digits = (int)log10(num) + 1; // Count the number of digits

    while (num > 0) {
        int digit = num % 10;
        sum += pow(digit, digits);
        num /= 10;
    }

    return sum == original_num; // Check if sum of powers equals the original number
}

// Function to check if a number is a perfect number
int is_perfect(int num) {
    int sum = 0;

    for (int i = 1; i <= num / 2; i++) {
        if (num % i == 0) {
            sum += i; // Add divisors
        }
    }

    return sum == num; // Check if sum of divisors equals the number
}

```

```c
#include <stdio.h>

// Recursive function to calculate power
double power(int base, int exponent) {
    // Base case: any number to the power of 0 is 1
    if (exponent == 0) {
        return 1;
    }
    // If the exponent is negative, calculate the positive power and take reciprocal
    else if (exponent < 0) {
        return 1 / power(base, -exponent);
    }
    // Recursive case
    else {
        return base * power(base, exponent - 1);
    }
}

int main() {
    int base, exponent;

    // Input base and exponent from the user
    printf("Enter base: ");
    scanf("%d", &base);
    printf("Enter exponent: ");
    scanf("%d", &exponent);

    // Calculate and display the result
    double result = power(base, exponent);
    printf("%d raised to the power of %d is: %.2f\n", base, exponent, result);

    return 0;
}

```