## Code

```c
#include <stdio.h>

void logicGates() {
    int a, b;

    printf("Enter a number (0 or 1) for NOT Gate: ");
    scanf("%d", &a);
    printf("NOT %d = %d\n", a, !a);


    printf("\nEnter two numbers (0 or 1) for AND Gate: ");
    scanf("%d %d", &a, &b);
    printf("%d AND %d = %d\n", a, b, a && b);


    printf("\nEnter two numbers (0 or 1) for OR Gate: ");
    scanf("%d %d", &a, &b);
    printf("%d OR %d = %d\n", a, b, a || b);


    printf("\nEnter two numbers (0 or 1) for NAND Gate: ");
    scanf("%d %d", &a, &b);
    printf("%d NAND %d = %d\n", a, b, !(a && b));


    printf("\nEnter two numbers (0 or 1) for NOR Gate: ");
    scanf("%d %d", &a, &b);
    printf("%d NOR %d = %d\n", a, b, !(a || b));
}

int main() {
    logicGates();
    return 0;
}
```

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

```c
// Function to convert decimal to binary
void decimalToBinary(int num) {
    int binary[32], i = 0;
    while (num > 0) {
        binary[i] = num % 2;
        num /= 2;
        i++;
    }
    printf("Binary: ");
    for (int j = i - 1; j >= 0; j--)
        printf("%d", binary[j]);
    printf("\n");
}

// Function to add two 16-bit binary numbers
void addBinaryNumbers() {
    int binary1[16], binary2[16], result[17], carry = 0;
    printf("Enter first 16-bit binary number: ");
    for (int i = 0; i < 16; i++) scanf("%1d", &binary1[i]);
    printf("Enter second 16-bit binary number: ");
    for (int i = 0; i < 16; i++) scanf("%1d", &binary2[i]);

    for (int i = 15; i >= 0; i--) {
        int sum = binary1[i] + binary2[i] + carry;
        result[i + 1] = sum % 2;
        carry = sum / 2;
    }
    result[0] = carry;

    printf("Sum: ");
    for (int i = 0; i < 17; i++) printf("%d", result[i]);
    printf("\n");
}

// Function to add (-124)₁₀ and (236)₁₀ and display in binary
void addSignedDecimals() {
    int num1 = -124, num2 = 236, sum = num1 + num2;
    printf("\nSum of %d and %d is %d\n", num1, num2, sum);
    decimalToBinary(sum);
}

// Function to convert decimal to hexadecimal
void decimalToHexadecimal(int num) {
    printf("Hexadecimal: %X\n", num);
}
```

```c
int main() {
    int choice, num;

    printf("Choose an option:\n");
    printf("1. Convert Decimal to Binary\n");
    printf("2. Add Two 16-bit Binary Numbers\n");
    printf("3. Add (-124) and (236) in Binary\n");
    printf("4. Convert Decimal to Hexadecimal\n");
    scanf("%d", &choice);

    switch (choice) {
    case 1:
        printf("Enter a decimal number: ");
        scanf("%d", &num);
        decimalToBinary(num);
        break;
    case 2:
        addBinaryNumbers();
        break;
    case 3:
        addSignedDecimals();
        break;
    case 4:
        printf("Enter a decimal number: ");
        scanf("%d", &num);
        decimalToHexadecimal(num);
        break;
    default:
        printf("Invalid choice!\n");
    }

    return 0;
}
```

```c
#include <stdio.h>

void intToBinary(int n, char *binary) {
    for (int i = 0; i < 32; i++) {
        binary[i] = (n & (1 << (31 - i))) ? '1' : '0';
    }
    binary[32] = '\0'; // Null-terminate the string
}

int main() {
    // Given numbers
    int num1 = -124;  // Decimal -124
    int num2 = 236;   // Decimal 236

    // Step 1: Convert num1 (-124) and num2 (236) to 32-bit 2's complement
binary format
    char binary_num1[33], binary_num2[33];
    intToBinary(num1, binary_num1);
    intToBinary(num2, binary_num2);

    // Step 2: Add the numbers
    int sum = num1 + num2;

    // Step 3: Convert the sum to binary format
    char binary_sum[33];
    intToBinary(sum, binary_sum);

    // Output the results
    printf("num1 = %d (binary: %s)\n", num1, binary_num1);
    printf("num2 = %d (binary: %s)\n", num2, binary_num2);
    printf("Sum = %d (binary: %s)\n", sum, binary_sum);

    return 0;
}
```

## Greatest

```

                AREA    GREATEST, CODE, READONLY
        ENTRY

        LDR     R0, =numbers
        LDR     R1, [R0]
        LDR     R2, =len
        MOV     R3, #1

check_loop
        CMP     R3, R2
        BEQ     done

        LDR     R4, [R0, R3, LSL #2]
        CMP     R1, R4
        MOVLT   R1, R4
        ADD     R3, R3, #1
        B       check_loop

done

        LDR     R5, =max_result
        STR     R1, [R5]


        B       done


        AREA    GREATEST, DATA, READWRITE
numbers  DCD     25, 10, 47, 36, 89, 58
len      DCD     6
max_result DCD   0

        END
```

## Smallest

```

        AREA s,CODE,READONLY
        ENTRY
        START
        MOV R5,#6
        LDR R1,=ARRAY
        LDR R2,[R1],#4
loop
        LDR R4,[R1],#4
        CMP R2,R4
        MOV R2,R4
        BLO loop1
loop1
        SUBS R5,R5,#1
        CMP R5,#0
        BNE loop
        LDR R4,=VALUE
        STR R2,[R4]

        NOP
        NOP
        NOP
ARRAY
        DCD 0x11111111
        DCD 0x22222222
        DCD 0x33333333
        DCD 0x44444444
        DCD 0x55555555
        DCD 0x66666666
        DCD 0x77777777


        AREA DATA1,DATA, READWRITE
VALUE DCD 0x0
        END
```

## Descending

```

                AREA Sort, CODE, READONLY
        ENTRY

        MOV R0, #10
        MOV R1, #20
        MOV R2, #15

        CMP R0, R1
        BGE skip1
        MOV R3, R0
        MOV R0, R1
        MOV R1, R3

skip1
        CMP R0, R2
        BGE skip2
        MOV R3, R0
        MOV R0, R2
        MOV R2, R3

skip2
        CMP R1, R2
        BGE done
        MOV R3, R1
        MOV R1, R2
        MOV R2, R3

done
        END
```

## Ascending

```
                AREA Sort, CODE, READONLY
        ENTRY

        MOV R0, #10     ; R0 = 10
        MOV R1, #20     ; R1 = 20
        MOV R2, #15     ; R2 = 15

        ; Compare R0 and R1
        CMP R0, R1
        BLE skip1        ; If R0 <= R1, skip swapping
        MOV R3, R0
        MOV R0, R1       ; Swap R0 and R1
        MOV R1, R3

skip1
        ; Compare R1 and R2
        CMP R1, R2
        BLE skip2        ; If R1 <= R2, skip swapping
        MOV R3, R1
        MOV R1, R2       ; Swap R1 and R2
        MOV R2, R3

skip2
        ; Compare R0 and R1 again
        CMP R0, R1
        BLE done         ; If R0 <= R1, sorting is complete
        MOV R3, R0
        MOV R0, R1       ; Swap R0 and R1
        MOV R1, R3

done
        END
```