Problem 1:

Write an ARM7 assembly program that involves a subroutine to subtract two numbers in two ways:

a) In the first way, assume that the operands are in the memory however your subroutine should pass its parameters via registers, and the subroutine should store the result back in the memory location just after the two operands.

b) In the second way, pass the parameters via memory and store the returned value back in the first memory location available just after the call.

Solution:

```
        AREA SubData, DATA, READWRITE
op1 DCD 5
op2 DCD 3
RES DCD 0

        AREA SubInst, CODE, READWRITE
        ENTRY
        LDR R1, op1
        LDR R2, op2
        BL Subtract
        B EXT

Subtract SUB R3, R1, R2
         STR R3, RES
         MOV PC, LR
EXT
        END
```

```
            AREA SubInst,CODE, READWRITE
            ENTRY
            BL Subtract
            DCD 5
            DCD 3
            DCD 0
            B EXT

Subtract LDR R1 , [LR]
             ADD LR, LR, #4
             LDR R2 , [LR]
             ADD LR, LR, #4
             SUB R3, R1, R2
             STR R3, [LR]
             ADD LR, LR, #4
             MOV PC, LR
EXT
            END
```

Problem 2:

Write an ARM7 assembly program that involves a subroutine to complement (1’s complement) a value and store its complement back in memory. Your parameter must be passed via memory and the returned complemented value must be stored in the first available memory location after the call.

Solution:

```
        AREA logical, CODE, READWRITE
        ENTRY
        BL COM
        DCD 0x33F1AA44
        DCD 0
        B EXT

COM      LDR R1,[LR]
        MVN R2, R1
        ADD LR,LR,#4
        STR R2,[LR]
        ADD LR,LR,#4
        MOV PC, LR
EXT
        END
```

Problem 3:

Write an ARM7 assembly program that involves a subroutine to sum up an array of elements. Your subroutine have 2 parameters that are passed via memory. The first parameter is the length of the array and the second parameter is the array. The summation returned must be stored in the first available memory location after the call.

Solution:

```
            AREA summation, CODE, READWRITE
            ENTRY
            BL SUM
            DCD 8
            DCD 1,2,10,15,3,6,4,0
            DCD 0
            B EXT

SUM LDR R0, [LR]
         ADD LR, LR, #4
         MOV R3, #0
         MOV R5, #0

LOP CMP R3, R0
       BEQ Done
       LDR R1, [LR]
       ADD R5, R5, R1
       ADD LR, LR, #4
       ADD R3, R3, #1
       B LOP

Done      STR R5, [LR]
            ADD LR, LR, #4
            MOV PC, LR
EXT
            END
```

Problem 4:

Write an ARM7 assembly program that involves a subroutine to search an array of elements for a specific number. Your subroutine have 3 parameters that are passed via memory. The first parameter is the length of the array, the second parameter is the element to search for and the third parameter is the array. If the element is found, your subroutine should store the value 1 in register R5, and if not found, the value in R5 is zero.

Solution:

```
        AREA searching, CODE, READWRITE
        ENTRY
        BL SRH
        DCD 8
        DCD 15
        DCD 1,2,10,15,3,6,4,0
        B EXT

SRH     LDR R0,[LR]
        ADD LR,LR, #4
        LDR R1,[LR]
        ADD LR, LR, #4
        MOV R3, #0
        MOV R5, #0

LOP     CMP R3, R0
        BEQ DONE
        LDR R2, [LR]
        ADD LR, LR, #4
        CMP R2, R1
        MOVEQ R5, #1
        ADD R3, R3, #1
        B LOP

DONE    MOV PC, LR
EXT
        END
```

Problem 5:

Write an ARM7 assembly program to evaluate the following equation $a \times b + c \times d$. Your program should involve two subroutines, one responsible for multiplying two numbers, and one responsible for addition of two numbers.

The multiplication subroutine should get its two parameters via memory, and the returned result should be stored in the first memory location available after the call, whereas the addition subroutine should store its returned value in the memory location after the call.

Solution:

```
           AREA subroutine2, CODE, READWRITE
           ENTRY
            BL MUL1
            DCD 5
            DCD 3
NUM1        DCD 0
            BL MUL1
            DCD 6
            DCD 4
NUM2        DCD 0
            BL ADD3
            DCD 0
            B EXT

MUL1        LDR R1,[LR]
            ADD LR, LR, #4
            LDR R2,[LR]
            ADD LR, LR, #4
            MUL R3, R1, R2
            STR R3, [LR]
            ADD LR, LR, #4
            MOV PC ,LR

ADD3 LDR R1, NUM1
            LDR R2, NUM2
            ADD R3, R1, R2
            STR R3, [LR]
           ADD LR, LR, #4
            MOV PC, LR
EXT
            END
```