Problem 1:

Write an assembly program that involves a subroutine to calculate the equation (A+B) – (C+D), given that A, B, C and D are four different numbers stored in the memory and passed to the subroutine as parameters. The subroutine should return a value and then the main program multiplies it by 2.

Solution:

```
.globl main
.data
num1: .word 6
num2: .word 4
num3: .word 5
num4: .word 3

.text
main:
        la $t0, num1
        lw $a0, 0($t0)
        la $t1, num2
        lw $a1, 0($t1)
        la $t2, num3
        lw $a2, 0($t2)
        la $t3, num4
        lw $a3, 0($t3)
        jal solve
        li $t4, 2
        mul $s3, $v0, $t4

        li $v0,10
        syscall

solve:
        add $s1, $a0, $a1
        add $s2, $a2, $a3
        sub $v0, $s1, $s2
        jr $ra
```

Problem 2:

Write an assembly program that involves a subroutine to calculate the 2’s complement of a value and store its complement back in memory. Your parameter should be first read from memory and the complemented value must be stored in the memory.

Solution:

```
.globl main
.data
num: .word 50
res: .word 0

.text
main:
        la $t0, num
        lw $a0, 0($t0)
        jal complement

        li $v0,10
        syscall

complement:
        not $t1,$a0
        addi $t1, 1
        la $t2, res
        sw $t2,0($t2)
        jr $ra
```

Problem 3:

Write an assembly program that involves a subroutine to sum up an array of elements. Your subroutine has 2 parameters. The first parameter is the array’s length, and the second is the array. The summation must be stored in a memory location.

Solution:

```
.globl main
.data
length: .word 8
arr: .word 60, 20, 10, 30, 5, 9, 7, 3
sum: .word 0

.text
main:
        la $t0, length
        lw $a0, 0($t0)
        la $a1, arr
        jal summation

        li $v0,10
        syscall

summation:
        loop:
         slt $t3, $t2, $a0
         beq $t3, $zero, Exit
         lw $s1, 0($a1)
         addi $a1, 4
         add $s2, $s2, $s1
         addi $t2, 1
         j loop

        Exit:
        la $t4, sum
        sw $s2,0($t4)
        jr $ra
```

Problem 4:

Write an assembly program that involves a subroutine to search an array of elements for a specific number. Your subroutine has 3 parameters. 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 $s5, and if not found, the value in $s5 is -1.

Solution:

```
.globl main
.data
length: .word 8
num: .word 30
arr: .word 60, 20, 10, 30, 5, 9, 7, 3

.text
main:
        la $t0, length
        lw $a0, 0($t0)
        la $a1, arr
        la $t4, num
        lw $a2, 0($t4)
        jal search

        li $v0,10
        syscall

search:
         li $s5, -1
        loop:
         slt $t3, $t2, $a0
         beq $t3, $zero, Exit
         lw $s1, 0($a1)
         addi $a1, 4
         beq $s1,$a2, equal
         addi $t2, 1
         j loop

        equal:
        li $s5,1
        Exit:
        jr $ra
```

Problem 5:

Write an assembly program which calls two subroutines, the first one to calculate the minimum element in array, the second subroutine is to get the maximum element of the same array, then the main program should calculate the difference between maximum and minimum and stores it in the memory. (Assume initially that the initial value of minimum is 1000, and the initial value of maximum is zero)

Solution:

```
.globl main
.data
length: .word 8
arr: .word 60, 20, 10, 30, 5, 9, 7, 3
diff: .word 0
.text
main:
            la $t0, length
            lw $a0, 0($t0)
            la $a1, arr
            jal minimum
            move $s4, $v0

            la $a1, arr
            jal maximum
            move $s5, $v0
            sub $s6,$s5,$s4
            la $t5, diff
            sw $s6, 0($t5)

            li $v0,10
            syscall

minimum:
             li $t4, 1000
            loop:
             slt $t3, $t2, $a0
             beq $t3, $zero, Exit
             lw $s1, 0($a1)
             addi $a1, 4
             blt $s1, $t4, min
            cnt:
             addi $t2, 1
             j loop
            min:
            move $t4, $s1
            move $v0, $s1
            j cnt
            Exit:
            jr $ra

maximum:
              li $t4, 0
              li $t2, 0
            loop2:
             slt $t3, $t2, $a0
             beq $t3, $zero, Exit2
             lw $s1, 0($a1)
             addi $a1, 4
             bgt $s1, $t4, max
             cnt2:
             addi $t2, 1
             j loop2
            max:
            move $t4, $s1
            move $v0, $s1
            j cnt2
            Exit2:
            jr $ra
```