Solution of Tutorial Series No. 4

Exercise 1 :

```
.data
array: .word 4, 8, 15, 16, 23, 42 # Example array
size: .word 6                  # Number of elements in the array
largest_gap_msg: .asciiz "The largest gap is: "

.text
.globl main

main:
   # Load array size and initialize variables
   la $t0, array         # Load base address of the array
   lw $t1, size          # Load size of the array into $t1
   li $t2, 0             # Largest gap initialized to 0

   # loop_1: Iterate through each element
   li $t3, 0             # loop_1 index i

loop_1:
   beq $t3, $t1, print_result # If i == size, exit loop
   lw $t4, 0($t0)        # Load array[i] into $t4
   addi $t5, $t3, 1      # Start inner loop at i + 1
   la $t6, array         # Reset base address of the array
   sll $t7, $t5, 2       # Compute offset for array[j]
   add $t6, $t6, $t7     # Point to array[j]

loop_2:
   beq $t5, $t1, next_outer # If j == size, go to next outer loop iteration
   lw $t8, 0($t6)        # Load array[j] into $t8
   sub $t9, $t8, $t4     # Calculate difference: array[j] - array[i]
   abs $t9, $t9          # Take the absolute value
   bgt $t9, $t2, update_gap # If gap > largest_gap, update it
   j next_inner

update_gap:
   move $t2, $t9         # Update largest gap

next_inner:
   addi $t5, $t5, 1      # Increment inner loop index
   addi $t6, $t6, 4      # Move to next element in array
   j loop_2

next_outer:
   addi $t3, $t3, 1      # Increment outer loop index
   addi $t0, $t0, 4      # Move to next element in array
```

```
    j loop_1

# Print the result
print_result:
    li $v0, 4           # Syscall: print string
    la $a0, largest_gap_msg
    syscall

    li $v0, 1           # Syscall: print integer
    move $a0, $t2       # Largest gap
    syscall

    li $v0, 10          # Exit syscall
    syscall
```

Exercise 2:

```
.data
T1: .word 4, 8, 7, 12     # Array T1
T2: .word 3, 6            # Array T2
n:  .word 4               # Size of T1
m:  .word 2               # Size of T2
schtroumpf_msg: .asciiz "The Schtroumpf value is: "

.text
.globl main

main:
    # Initialize
    la $t0, T1            # Base address of T1
    la $t1, T2            # Base address of T2
    lw $t2, n             # Size of T1
    lw $t3, m             # Size of T2
    li $t4, 0             # Schtroumpf accumulator

loop_1:
    li $t5, 0             # loop_1 index
    la $t6, T2            # Reset base address of T2

loop_2:
    beq $t5, $t3, next_outer # If loop_2 complete, go to outer loop
    lw $t7, 0($t0)        # Load current T1 element
    lw $t8, 0($t6)        # Load current T2 element
    mul $t9, $t7, $t8     # Multiply
    add $t4, $t4, $t9     # Add to Schtroumpf
    addi $t5, $t5, 1      # Increment inner loop index
    addi $t6, $t6, 4      # Move to the next T2 element
    j loop_2
```

```
next_outer:
   addi $t2, $t2, -1      # Decrement outer loop count
   addi $t0, $t0, 4       # Move to the next T1 element
   bnez $t2, loop_1   # If T1 not exhausted, repeat

   # Print Result
   li $v0, 4            # Print message
   la $a0, schtroumpf_msg
   syscall

   li $v0, 1            # Print Schtroumpf value
   move $a0, $t4
   syscall

   li $v0, 10           # Exit
   syscall
```

Exercise 3 :

```
.data
grades: .space 12          # Space for 10 grades
size: .word 3              # Number of grades
msg_input: .asciiz "Enter a grade: "
msg_result: .asciiz "Grades above the average: "

.text
.globl main

main:
   # Input grades
   la $t0, grades         # Base address of grades
   lw $t1, size           # Number of grades
   li $t2, 0              # Sum of grades

input_loop:
   beqz $t1, compute_avg   # If all grades entered, compute average
   li $v0, 4              # Print input message
   la $a0, msg_input
   syscall

   li $v0, 5              # Read grade
   syscall
   sw $v0, 0($t0)         # Store grade
   add $t2, $t2, $v0      # Add to sum
   addi $t0, $t0, 4       # Move to next grade
   subi $t1, $t1, 1       # Decrement counter
   j input_loop
```

```
compute_avg:
   lw $t1, size          # Reload size
   div $t2, $t1          # Compute average
   mflo $t3              # Store average in $t3

count_above_avg:
   la $t0, grades        # Reset base address
   li $t4, 0             # Counter for grades above average
   lw $t1, size          # Reload size

count_loop:
   beqz $t1, print_result  # If all grades checked, print result
   lw $t5, 0($t0)        # Load current grade
   bgt $t5, $t3, increment # If grade > average, increment counter
   addi $t0, $t0, 4      # Move to next grade
   subi $t1, $t1, 1      # Decrement counter
   j count_loop

increment:
   addi $t4, $t4, 1      # Increment counter
   addi $t0, $t0, 4      # Move to next grade
   subi $t1, $t1, 1      # Decrement counter
   j count_loop

print_result:
   li $v0, 4             # Print message
   la $a0, msg_result
   syscall

   li $v0, 1             # Print number of grades above average
   move $a0, $t4
   syscall

   li $v0, 10            # Exit
   syscall
```