Solution of Tutorial Series No. 3

Exercise No. 1: Comparing Two Numbers

```
    .data
promptA: .asciiz "Enter the first number (A): "
promptB: .asciiz "Enter the second number (B): "
output:  .asciiz "The larger number is: "

    .text
    .globl main
main:
    # Read integer A
    li $v0, 4
    la $a0, promptA
    syscall

    li $v0, 5
    syscall
    move $t0, $v0  # $t0 stores A

    # Read integer B
    li $v0, 4
    la $a0, promptB
    syscall

    li $v0, 5
    syscall
    move $t1, $v0  # $t1 stores B

    # Compare A and B
    bge $t0, $t1, A_is_larger
    move $t2, $t1  # If B is larger, store B in $t2
    j print_result

A_is_larger:
    move $t2, $t0  # If A is larger, store A in $t2

print_result:
    # Print result
    li $v0, 4
    la $a0, output
    syscall

    li $v0, 1
    move $a0, $t2
    syscall
```

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

Exercise No. 2: Calculating the Factorial of a Number

```
    .data
promptN: .asciiz "Enter a positive integer (N): "
output:  .asciiz "The factorial is: "

    .text
    .globl main
main:
    # Read integer N
    li $v0, 4
    la $a0, promptN
    syscall

    li $v0, 5
    syscall
    move $t0, $v0  # $t0 stores N

    # Initialize factorial calculation
    li $t1, 1      # $t1 will store the factorial result
    li $t2, 1      # Counter variable starting from 1

factorial_loop:
    bgt $t2, $t0, end_factorial  # If counter > N, exit loop
    mul $t1, $t1, $t2          # Multiply factorial result by counter
    addi $t2, $t2, 1           # Increment counter
    j factorial_loop

end_factorial:
    # Print result
    li $v0, 4
    la $a0, output
    syscall

    li $v0, 1
    move $a0, $t1
    syscall

    li $v0, 10
    syscall  # Exit
```

Exercise No. 3: Calculating the Sum and Average of a Series of Numbers

```
    .data
promptN: .asciiz "Enter the number of values (N): "
promptValue: .asciiz "Enter a value: "
sum_output: .asciiz "The sum is: "
avg_output: .asciiz "The average is: "

    .text
    .globl main
main:
    # Read integer N
    li $v0, 4
    la $a0, promptN
    syscall

    li $v0, 5
    syscall
    move $t0, $v0  # $t0 stores N

    # Initialize sum
    li $t1, 0      # Sum initialized to 0
    li $t2, 0      # Counter initialized to 0

sum_loop:
    bge $t2, $t0, end_sum  # If counter >= N, exit loop

    # Prompt for value
    li $v0, 4
    la $a0, promptValue
    syscall

    li $v0, 5
    syscall
    add $t1, $t1, $v0  # Add value to sum
    addi $t2, $t2, 1   # Increment counter
    j sum_loop

end_sum:
    # Calculate average
    div $t1, $t0       # Divide sum by N
    mflo $t3          # Store result in $t3 (average)

    # Print sum
    li $v0, 4
    la $a0, sum_output
    syscall
```

```
    li $v0, 1
    move $a0, $t1
    syscall

    # Print average
    li $v0, 4
    la $a0, avg_output
    syscall

    li $v0, 1
    move $a0, $t3
    syscall

    li $v0, 10
    syscall  # Exit
```

Exercise No. 4: Calculating the PGCD (Euclidean Algorithm)

```
    .data
promptA: .asciiz "Enter the first number (A): "
promptB: .asciiz "Enter the second number (B): "
output:  .asciiz "The GCD is: "

    .text
    .globl main
main:
    # Read integer A
    li $v0, 4
    la $a0, promptA
    syscall

    li $v0, 5
    syscall
    move $t0, $v0  # $t0 stores A

    # Read integer B
    li $v0, 4
    la $a0, promptB
    syscall

    li $v0, 5
    syscall
    move $t1, $v0  # $t1 stores B

gcd_loop:
    beq $t1, 0, end_gcd  # If B == 0, GCD found
    div $t0, $t1         # Divide A by B
    mfhi $t2             # Get remainder
```

```
    move $t0, $t1       # Set A = B
    move $t1, $t2       # Set B = remainder
    j gcd_loop

end_gcd:
    # Print result
    li $v0, 4
    la $a0, output
    syscall

    li $v0, 1
    move $a0, $t0
    syscall

    li $v0, 10
    syscall  # Exit
```