Lab Solution Series No. 2

Exercise No. 1: Calculating the Perimeter and Area of a Rectangle

```
.data
prompt_length: .asciiz "Enter the length of the rectangle: "
prompt_width:  .asciiz "Enter the width of the rectangle: "
result_perimeter: .asciiz "The perimeter of the rectangle is: "
result_area: .asciiz "The area of the rectangle is: "

.text
.globl main
main:
   # Read length
   li $v0, 4               # System call for print string
   la $a0, prompt_length       # Load address of prompt
   syscall                 # Print prompt
   li $v0, 5               # System call for read integer
   syscall                 # Read length
   move $t0, $v0               # Store length in $t0

   # Read width
   li $v0, 4               # System call for print string
   la $a0, prompt_width        # Load address of prompt
   syscall                 # Print prompt
   li $v0, 5               # System call for read integer
   syscall                 # Read width
   move $t1, $v0               # Store width in $t1

   # Calculate perimeter: 2 * (length + width)
   add $t2, $t0, $t1           # t2 = length + width
   sll $t2, $t2, 1           # t2 = t2 * 2 (shift left by 1)

   # Display perimeter
   li $v0, 4               # System call for print string
   la $a0, result_perimeter    # Load address of result_perimeter
   syscall                 # Print "The perimeter of the rectangle is: "
   li $v0, 1               # System call for print integer
   move $a0, $t2               # Load perimeter result
   syscall                 # Print perimeter

   # Calculate area: length * width
   mul $t3, $t0, $t1           # t3 = length * width

   # Display area
   li $v0, 4               # System call for print string
   la $a0, result_area         # Load address of result_area
```

```
    syscall                 # Print "The area of the rectangle is: "
    li $v0, 1               # System call for print integer
    move $a0, $t3           # Load area result
    syscall                 # Print area

    # Exit program
    li $v0, 10              # System call for exit
    syscall
```

Exercise No. 2: Calculating the Average of Three Grades

```
.data
prompt_grade1: .asciiz "Enter Grade 1: "
prompt_grade2: .asciiz "Enter Grade 2: "
prompt_grade3: .asciiz "Enter Grade 3: "
result_average: .asciiz "The average grade is: "

.text
.globl main
main:
    # Read Grade 1
    li $v0, 4               # System call for print string
    la $a0, prompt_grade1   # Load address of prompt
    syscall                 # Print prompt
    li $v0, 5               # System call for read integer
    syscall                 # Read Grade 1
    move $t0, $v0           # Store Grade 1 in $t0

    # Read Grade 2
    li $v0, 4               # System call for print string
    la $a0, prompt_grade2   # Load address of prompt
    syscall                 # Print prompt
    li $v0, 5               # System call for read integer
    syscall                 # Read Grade 2
    move $t1, $v0           # Store Grade 2 in $t1

    # Read Grade 3
    li $v0, 4               # System call for print string
    la $a0, prompt_grade3   # Load address of prompt
    syscall                 # Print prompt
    li $v0, 5               # System call for read integer
    syscall                 # Read Grade 3
    move $t2, $v0           # Store Grade 3 in $t2

    # Calculate average: (Grade1 + Grade2 + Grade3) / 3
    add $t3, $t0, $t1       # t3 = Grade1 + Grade2
    add $t3, $t3, $t2       # t3 = t3 + Grade3
```

```
    li $t4, 3               # Set divisor to 3
    div $t3, $t4            # Divide by 3
    mflo $t5                # Move result (quotient) to $t5

    # Display average
    li $v0, 4               # System call for print string
    la $a0, result_average  # Load address of result_average
    syscall                 # Print "The average grade is: "
    li $v0, 1               # System call for print integer
    move $a0, $t5           # Load average result
    syscall                 # Print average

    # Exit program
    li $v0, 10              # System call for exit
    syscall
```

Exercise No. 3: Calculating the Power of a Number

```
.data
prompt_base: .asciiz "Enter the base: "
prompt_exponent: .asciiz "Enter the exponent: "
result_power: .asciiz "The result of base^exponent is: "

.text
.globl main
main:
    # Read base
    li $v0, 4               # System call for print string
    la $a0, prompt_base     # Load address of prompt
    syscall                 # Print prompt
    li $v0, 5               # System call for read integer
    syscall                 # Read base
    move $t0, $v0           # Store base in $t0

    # Read exponent
    li $v0, 4               # System call for print string
    la $a0, prompt_exponent # Load address of prompt
    syscall                 # Print prompt
    li $v0, 5               # System call for read integer
    syscall                 # Read exponent
    move $t1, $v0           # Store exponent in $t1

    # Initialize result to 1 (base^0 = 1)
    li $t2, 1               # Set initial result to 1

power_loop:
    beq $t1, 0, end_power   # If exponent is 0, exit loop
```

```
    mul $t2, $t2, $t0          # Multiply result by base
    sub $t1, $t1, 1            # Decrement exponent by 1
    j power_loop               # Repeat loop

end_power:
    # Display result
    li $v0, 4                  # System call for print string
    la $a0, result_power       # Load address of result_power
    syscall                    # Print "The result of base^exponent is: "
    li $v0, 1                  # System call for print integer
    move $a0, $t2              # Load result
    syscall                    # Print result

    # Exit program
    li $v0, 10                 # System call for exit
    syscall
```

Exercise No. 4: Finding the Largest Number in a Series

```
.data
prompt_N: .asciiz "Enter the number of values (N): "
prompt_value: .asciiz "Enter a value: "
result_largest: .asciiz "The largest number is: "

.text
.globl main
main:
    # Lire N
    li $v0, 4                  # System call for print string
    la $a0, prompt_N           # Load address of prompt_N
    syscall                    # Print "Enter the number of values (N): "
    li $v0, 5                  # System call for read integer
    syscall                    # Read N
    move $t0, $v0              # Store N in $t0 (N is the count of numbers)

    # Lire le premier nombre et l'utiliser comme le plus grand actuel
    li $v0, 4                  # System call for print string
    la $a0, prompt_value       # Load address of prompt_value
    syscall                    # Print "Enter a value: "
    li $v0, 5                  # System call for read integer
    syscall                    # Read the first value
    move $t1, $v0              # Store the first value in $t1 as the initial largest number

    # Décrémenter N car le premier nombre a déjà été lu
    sub $t0, $t0, 1

compare_loop:
```

```
    # Vérifier si N > 0
    blez $t0, end_compare        # If N <= 0, end loop

    # Lire la prochaine valeur
    li $v0, 4                    # System call for print string
    la $a0, prompt_value         # Load address of prompt_value
    syscall                      # Print "Enter a value: "
    li $v0, 5                    # System call for read integer
    syscall                      # Read the next value
    move $t2, $v0                # Store the value in $t2

    # Comparer avec le plus grand actuel
    bgt $t2, $t1, update_largest  # If value > current largest, update largest
    j skip_update                # Otherwise, skip updating

update_largest:
    move $t1, $t2                # Update the largest number to the current value

skip_update:
    # Décrémenter N et répéter
    sub $t0, $t0, 1              # Decrement N
    j compare_loop               # Repeat the loop

end_compare:
    # Afficher le plus grand nombre
    li $v0, 4                    # System call for print string
    la $a0, result_largest       # Load address of result_largest
    syscall                      # Print "The largest number is: "
    li $v0, 1                    # System call for print integer
    move $a0, $t1                # Load largest number
    syscall                      # Print largest

    # Quitter le programme
    li $v0, 10                   # Exit
    syscall
```

Exercise No. 5: Prime Number Verification

```
.data
prompt_N: .asciiz "Enter an integer N: "
result_prime: .asciiz "N is a prime number."
result_not_prime: .asciiz "N is not a prime number."

.text
.globl main
main:
    # Read N
```

```
    li $v0, 4               # Print string
    la $a0, prompt_N        # Load address of prompt_N
    syscall                 # Print "Enter an integer N: "
    li $v0, 5               # Read integer
    syscall                 # Read N
    move $t0, $v0           # Store N in $t0

    # Check if N <= 1 (not prime if N <= 1)
    li $t1, 1               # Load 1 into $t1
    ble $t0, $t1, not_prime     # If N <= 1, jump to not_prime

    # Initialize variables for checking divisibility
    li $t2, 2               # Start divisor at 2

prime_check_loop:
    mul $t3, $t2, $t2       # Calculate t2 * t2 (check if divisor^2 > N)
    bgt $t3, $t0, is_prime      # If divisor^2 > N, N is prime

    # Check if N is divisible by current divisor
    div $t0, $t2            # Divide N by divisor
    mfhi $t3                # Get remainder
    beq $t3, 0, not_prime       # If remainder is 0, N is not prime

    # Increment divisor and repeat
    addi $t2, $t2, 1        # Increment divisor
    j prime_check_loop          # Repeat the loop

is_prime:
    # Display prime message
    li $v0, 4               # Print string
    la $a0, result_prime        # Load address of result_prime
    syscall                 # Print "N is a prime number"
    j end_program

not_prime:
    # Display not prime message
    li $v0, 4               # Print string
    la $a0, result_not_prime    # Load address of result_not_prime
    syscall                 # Print "N is not a prime number"

end_program:
    # Exit program
    li $v0, 10              # Exit
    syscall
```