Ruby Lab Assessment – 3

Suryakumar P 21MIS1146

1. Write a ruby code using the following keywords yield, lambda and procs.

Code:

```ruby
# a method that uses yield
def greet_with_yield
  puts "Inside the method before the block."
  yield if block_given?
  puts "Inside the method after the block."
end

greet_with_yield do
  puts "Hello from the block!"
end

puts "\n LAMBDA AND PROC \n"

# a lambda that returns a greeting message
greet_lambda = lambda { |name| "Hello, #{name} from the lambda!" }

puts greet_lambda.call("Alice")


# a proc that returns a farewell message
farewell_proc = Proc.new { |name| "Goodbye, #{name} from the proc!" }

puts farewell_proc.call("Bob")
```

Output:

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3> ruby .\yield.rb
Inside the method before the block.
Hello from the block!
Inside the method after the block.

 LAMBDA AND PROC
Hello, Alice from the lambda!
Goodbye, Bob from the proc!
```

2. Write a ruby programming using Modules concept.

Code:

```ruby
module MathOperations
  def add(a, b)
    a + b
  end

  def subtract(a, b)
    a - b
  end

  def multiply(a, b)
    a * b
  end

  def divide(a, b)
    return "Division by zero error" if b == 0
    a / b
  end
end

class Calculator
  include MathOperations
end

calc = Calculator.new

puts calc.add(5, 3)
puts calc.subtract(5, 3)
puts calc.multiply(5, 3)
puts calc.divide(5, 0)
```

Output:

```
OUTPUT   TERMINAL   PORTS   AZURE   DEBUG CONSOLE   pwsh

PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3> ruby .\module.rb
8
2
15
Division by zero error
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3>
```

3. Write a ruby programming using Mixins concept.

Code:

```ruby
module Walkable
   def walk
     puts "#{self.class} is walking."
   end
 end

 class Human
   include Walkable
 end

 class Dog
   include Walkable
 end

 class Robot
   include Walkable
 end

 human = Human.new
 dog = Dog.new
 robot = Robot.new

 human.walk
 dog.walk
 robot.walk
```

Output:

```
OUTPUT    TERMINAL    PORTS    AZURE    DEBUG CONSOLE    pwsh

PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3> ruby .\mixins.rb
Human is walking.
Dog is walking.
Robot is walking.
```

4. Write a ruby programming using Reflection concept.

Code:

```ruby
class Person

    attr_accessor :name, :age

    def initialize(name, age)
      @name = name
      @age = age
    end

    def greet
      "Hello, my name is #{@name} and I am #{@age} years old."
    end
  end

  person = Person.new("Surya", 20)

  puts "Class: #{person.class}"

  puts "\nInstance Variables:"
  puts person.instance_variables

  puts "\nName (using reflection): #{person.instance_variable_get(:@name)}"

  person.instance_variable_set(:@name, "Priyanka")
  puts "\nUpdated Name (using reflection): #{person.name}"

  puts "\nDoes person respond to 'greet'? #{person.respond_to?(:greet)}"
  puts "\nGreet method output: #{person.greet}"
```

Output:

```
ramming\Week4\Lab3> ruby .\reflection.rb
Class: Person

Instance Variables:
@name
@age

Name (using reflection): Surya

Updated Name (using reflection): Priyanka

Does person respond to 'greet'? true

Greet method output: Hello, my name is Priyanka and I am 20 years old.
```

5. Write a ruby programming using Meta-programming concept.

Code:

```ruby
class DynamicAttributes
    def initialize(attributes = {})
      attributes.each do |key, value|
        self.class.send(:define_method, key) do
          instance_variable_get("@#{key}")
        end

        self.class.send(:define_method, "#{key}=") do |val|
          instance_variable_set("@#{key}", val)
        end

        instance_variable_set("@#{key}", value)
      end
    end
  end

  person = DynamicAttributes.new(name: "Surya", age: 20)

  puts person.name
  puts person.age

  person.name = "Priyanka"
  person.age = 29

  puts person.name
  puts person.age
```

Output:

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3> ruby .\meta.rb
Surya
20
Priyanka
29
```

6. Create an array a=[1,2,3,4,5,6], and perform the following:

- Different ways to access the array elements
- Five different methods associated with array.
- Different ways to add and delete an element of an array.
- Introduce two new arrays and perform intersection, concatenation, difference.
- Perform a binary search using array a.

Code:

```
# Create an array
a = [1, 2, 3, 4, 5, 6]

# a. Different ways to access the array elements
puts "Accessing elements:"
puts "First element: #{a[0]}"
puts "Third element: #{a[2]}"
puts "Last element: #{a[-1]}"
puts "Range of elements (index 1 to 3): #{a[1..3]}"
puts "Using fetch method: #{a.fetch(2)}"

# b. Five different methods associated with array
puts "\nArray methods:"
puts "Length of array: #{a.length}"
puts "Reversed array: #{a.reverse}"
puts "Array includes 3? #{a.include?(3)}"
puts "Joined array into string: #{a.join('-')}"
puts "Sum of all elements: #{a.sum}"

# c. Different ways to add and delete an element of an array
puts "\nAdding and deleting elements:"
a.push(7)          # Adding an element to the end
puts "Array after push(7): #{a}"

a << 8             # Another way to add an element to the end
puts "Array after << 8: #{a}"

a.unshift(0)       # Adding an element to the beginning
puts "Array after unshift(0): #{a}"

a.insert(3, 10)    # Adding an element at index 3
puts "Array after insert(3, 10): #{a}"

a.pop              # Removing the last element
puts "Array after pop: #{a}"
```

```ruby
a.shift            # Removing the first element
puts "Array after shift: #{a}"

a.delete_at(2)     # Removing an element at index 2
puts "Array after delete_at(2): #{a}"

a.delete(10)       # Removing a specific element by value
puts "Array after delete(10): #{a}"

# d. Introduce two new arrays and perform intersection, concatenation, difference
b = [4, 5, 6, 7, 8]
c = [1, 2, 9, 10]

puts "\nArray b: #{b}"
puts "Array c: #{c}"

intersection = a & b
puts "Intersection of a and b: #{intersection}"

concatenation = a + c
puts "Concatenation of a and c: #{concatenation}"

difference = a - b
puts "Difference between a and b: #{difference}"

# e. Perform a binary search using array a
# Ensure the array is sorted
a.sort!
puts "\nSorted array a: #{a}"

# Binary search for the element '4'
def binary_search(arr, target)
  low = 0
  high = arr.length - 1

  while low <= high
    mid = (low + high) / 2
    if arr[mid] == target
      return "Element #{target} found at index #{mid}"
    elsif arr[mid] < target
      low = mid + 1
    else
      high = mid - 1
    end
  end
```

```
  return "Element #{target} not found in array"
end

puts binary_search(a, 4)
puts binary_search(a, 10)
```

Output:

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3> ruby .\array.rb
Accessing elements:
First element: 1
Third element: 3
Last element: 6
Range of elements (index 1 to 3): [2, 3, 4]
Using fetch method: 3

Array methods:
Length of array: 6
Reversed array: [6, 5, 4, 3, 2, 1]
Array includes 3? true
Joined array into string: 1-2-3-4-5-6
Sum of all elements: 21

Adding and deleting elements:
Array after push(7): [1, 2, 3, 4, 5, 6, 7]
Array after << 8: [1, 2, 3, 4, 5, 6, 7, 8]
Array after unshift(0): [0, 1, 2, 3, 4, 5, 6, 7, 8]
Array after insert(3, 10): [0, 1, 2, 10, 3, 4, 5, 6, 7, 8]
Array after pop: [0, 1, 2, 10, 3, 4, 5, 6, 7]
Array after shift: [1, 2, 10, 3, 4, 5, 6, 7]
Array after delete_at(2): [1, 2, 3, 4, 5, 6, 7]
Array after delete(10): [1, 2, 3, 4, 5, 6, 7]

Array b: [4, 5, 6, 7, 8]
Array c: [1, 2, 9, 10]
Intersection of a and b: [4, 5, 6, 7]
Concatenation of a and c: [1, 2, 3, 4, 5, 6, 7, 1, 2, 9, 10]
Difference between a and b: [1, 2, 3]

Sorted array a: [1, 2, 3, 4, 5, 6, 7]
Element 4 found at index 3
Element 10 not found in array
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming\Lab\Ruby-Prog
ramming\Week4\Lab3>
```

3.1

```
#Library Catalog:
#Code:


library_catalog = {}


def add_book_to_catalog(catalog)
  puts "Enter the book's title:"
  title = gets.chomp

  puts "Enter the author's name:"
  author = gets.chomp

  puts "Enter the genre:"
  genre = gets.chomp

  puts "Enter the publication year:"
  year = gets.chomp.to_i

  id = catalog.size + 1

  catalog[id] = { title: title, author: author, genre: genre, year: year }
end

def books_published_after(catalog, year)
  catalog.select { |id, book| book[:year] > year }
end

loop do
  add_book_to_catalog(library_catalog)

  puts "Do you want to add another book? (yes/no)"
  answer = gets.chomp.downcase
  break if answer != 'yes'
end

puts "Enter the year to search for books published after:"
search_year = gets.chomp.to_i

searched_books = books_published_after(library_catalog, search_year)
```

```ruby
puts "Books published after #{search_year}:"
searched_books.each do |id, book|
  puts "Title: #{book[:title]}, Author: #{book[:author]}, Genre: #{book[:genre]},
Year: #{book[:year]}"
end
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    powershell

PS D:\21MIS1146\Week4\Lab3_1> ruby .\library_catalog.rb
Enter the book's title:
Harry Potter 1
Enter the author's name:
JK Rowling
Enter the genre:
Fiction
Enter the publication year:
1990
Do you want to add another book? (yes/no)
yes
Enter the book's title:
Vampire Diary
Enter the author's name:
Priyanka
Enter the genre:
Fiction
Enter the publication year:
2014
Do you want to add another book? (yes/no)
no
Enter the year to search for books published after:
2000
Books published after 2000:
Title: Vampire Diary, Author: Priyanka, Genre: Fiction, Year: 2014
PS D:\21MIS1146\Week4\Lab3_1>
```

Employee Database Management:

Code:

```
employee_database = {}

def add_employee_to_database(database)
  puts "Enter the employee's name:"
  name = gets.chomp

  puts "Enter the employee's department:"
  department = gets.chomp

  puts "Enter the employee's salary:"
  salary = gets.chomp.to_f

  id = database.size + 1

  database[id] = { name: name, department: department, salary: salary }
end

def highest_paid_employee(database)
  database.max_by { |id, employee| employee[:salary] }
end

loop do
  add_employee_to_database(employee_database)

  puts "Do you want to add another employee? (yes/no)"
  answer = gets.chomp.downcase
  break if answer != 'yes'
end
highest_paid = highest_paid_employee(employee_database)

if highest_paid
  id, details = highest_paid
  puts "The highest-paid employee is:"
  puts "Name: #{details[:name]}"
  puts "Department: #{details[:department]}"
  puts "Salary: Rs.#{details[:salary]}"
else
  puts "No employees found in the database."
end
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    powershell

PS D:\21MIS1146\Week4\Lab3_1> ruby .\employee_data.rb
Enter the employee's name:
Suryakumar
Enter the employee's department:
Developer
Enter the employee's salary:
90000
Do you want to add another employee? (yes/no)
yes
Enter the employee's name:
Priyanka
Enter the employee's department:
Tester
Enter the employee's salary:
80000
Do you want to add another employee? (yes/no)

The highest-paid employee is:
Name: Suryakumar
Department: Developer
Salary: Rs.90000.0
PS D:\21MIS1146\Week4\Lab3_1>
```

Market Place

Code:

```
marketplace = {}

def add_product_to_marketplace(marketplace)
  puts "Enter the product name:"
  name = gets.chomp

  puts "Enter the product price:"
  price = gets.chomp.to_f

  puts "Enter the product quantity:"
  quantity = gets.chomp.to_i


  id = marketplace.size + 1


  marketplace[id] = { name: name, price: price, quantity: quantity }
end

def total_value_of_products(marketplace)
  marketplace.sum { |id, product| product[:price] * product[:quantity] }
end

loop do
  add_product_to_marketplace(marketplace)

  puts "Do you want to add another product? (yes/no)"
  answer = gets.chomp.downcase
  break if answer != 'yes'
end

total_value = total_value_of_products(marketplace)
puts "The total value of all products in the marketplace is: Rs. #{total_value}"
```

Output:

```
PROBLEMS    OUTPUT    TERMINAL    ...                    powershell

PS D:\21MIS1146\Week4\Lab3_1> ruby .\market_place.rb
Enter the product name:
Laptop
Enter the product price:
80000
Enter the product quantity:
10
Do you want to add another product? (yes/no)
yes
Enter the product name:
Mouse
Enter the product price:
600
Enter the product quantity:
50
Do you want to add another product? (yes/no)
yes
Enter the product name:
Mechanical Keyboard
Enter the product price:
2500
Enter the product quantity:
25
Do you want to add another product? (yes/no)
no
The total value of all products in the marketplace is: Rs. 892500.0
PS D:\21MIS1146\Week4\Lab3_1>
```

Student Grade:

Code:

```
def grade_to_points(grade)

  case grade
  when 'S' then 10
  when 'A' then 9
  when 'B' then 8
  when 'C' then 7
  when 'D' then 6
  when 'F' then 0
  else 0
  end
end

def calculate_cgpa(students_grades)
  students_grades.each do |student, grades|
    total_points = grades.map { |grade| grade_to_points(grade) }.sum
    cgpa = total_points.to_f / grades.size

    puts "#{student}'s CGPA: #{cgpa.round(2)}"
  end
end

def get_student_grades
  students_grades = {}

  puts "Enter the number of students:"
  number_of_students = gets.chomp.to_i

  number_of_students.times do
    puts "Enter the student's name:"
    student_name = gets.chomp

    puts "Enter the grades for #{student_name} (separated by spaces):"
    grades = gets.chomp.split

    students_grades[student_name] = grades
  end

  students_grades
end

students_grades = get_student_grades
```

```
calculate_cgpa(students_grades)
```

Output:

```
PROBLEMS    OUTPUT    TERMINAL    ...                    powershell

PS D:\21MIS1146\Week4\Lab3_1> ruby student_grade.rb
Enter the number of students:
3
Enter the student's name:
Surya
Enter the grades for Surya (separated by spaces):
S A S A S
Enter the student's name:
Priyanka
Enter the grades for Priyanka (separated by spaces):
S S A S S
Enter the student's name:
Rajeev
Enter the grades for Rajeev (separated by spaces):
S S S B S
Surya's CGPA: 9.6
Priyanka's CGPA: 9.8
Rajeev's CGPA: 9.6
PS D:\21MIS1146\Week4\Lab3_1>
```

Music Player:

Code:

```
playlist = {}

def add_song_to_playlist(playlist)
  puts "Enter the song title:"
  title = gets.chomp

  puts "Enter the artist name:"
  artist = gets.chomp

  puts "Enter the genre:"
  genre = gets.chomp

  id = playlist.size + 1

  playlist[id] = { title: title, artist: artist, genre: genre }
end

def shuffle_and_play(playlist)
  shuffled_playlist = playlist.keys.shuffle

  puts "Playing songs in random order:"
  shuffled_playlist.each do |id|
    song = playlist[id]
    puts "Now playing: '#{song[:title]}' by #{song[:artist]} [#{song[:genre]}]"
  end
end

loop do
  add_song_to_playlist(playlist)

  puts "Do you want to add another song? (yes/no)"
  answer = gets.chomp.downcase
  break if answer != 'yes'
end

shuffle_and_play(playlist)
```

Output:

```
PS D:\21MIS1146\Week4\Lab3_1> ruby .\music_player.rb
Enter the song title:
Monster
Enter the artist name:
Justin Beiber
Enter the genre:
Pop
Do you want to add another song? (yes/no)
yes
Enter the song title:
Starboy
Enter the artist name:
Weekend
Enter the genre:
LoFi
Do you want to add another song? (yes/no)
yes
Enter the song title:
Faded
Enter the artist name:
Alan Walker
Enter the genre:
Pop
Do you want to add another song? (yes/no)
no
Playing songs in random order:
Now playing: 'Faded' by Alan Walker [Pop]
Now playing: 'Monster' by Justin Beiber [Pop]
Now playing: 'Starboy' by Weekend [LoFi]
PS D:\21MIS1146\Week4\Lab3_1>
```

Q1.

Code:

```
require 'date'

# Generate a year's worth of random temperature data
def generate_temperature_data
  (1..365).map { rand(15..35) }
end

# Calculate the average temperature for the year
def average_temperature(temps)
  temps.sum / temps.size.to_f
end

# Find the hottest and coldest days of the year
def temperature_extremes(temps)
  hottest_day = temps.index(temps.max) + 1
  coldest_day = temps.index(temps.min) + 1
  [temps.max, hottest_day, temps.min, coldest_day]
end

# Calculate the average temperature for each month
def monthly_average(temps)
  days_in_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
  month_starts = days_in_month.each_with_index.map { |days, i|
days_in_month[0...i].sum }
  averages = []

  month_starts.each_with_index do |start_day, i|
    month_temps = temps[start_day, days_in_month[i]]
    averages << month_temps.sum / month_temps.size.to_f
  end

  averages
end

# Find the length of the longest heat wave
def longest_heat_wave(temps)
  max_length = 0
  current_length = 0

  temps.each do |temp|
    if temp > 30
```

```
      current_length += 1
      max_length = [max_length, current_length].max
    else
      current_length = 0
    end
  end

  max_length
end

# Find the length of the longest cold spell
def longest_cold_spell(temps)
  max_length = 0
  current_length = 0

  temps.each do |temp|
    if temp < 20
      current_length += 1
      max_length = [max_length, current_length].max
    else
      current_length = 0
    end
  end

  max_length
end

# Find the month with the highest average temperature
def hottest_month(temps)
  month_avgs = monthly_average(temps)
  hottest_month_index = month_avgs.index(month_avgs.max)
  hottest_month_index + 1 # months are 1-indexed
end

# Main Execution
temps = generate_temperature_data

puts "Average Temperature for the Year: #{average_temperature(temps).round(2)}°C"

hottest_temp, hottest_day, coldest_temp, coldest_day =
temperature_extremes(temps)
puts "Hottest Temperature: #{hottest_temp}°C on Day #{hottest_day}"
puts "Coldest Temperature: #{coldest_temp}°C on Day #{coldest_day}"

monthly_avgs = monthly_average(temps)
```

```
puts "Monthly Averages: #{monthly_avgs.map { |avg| avg.round() }.join(', ')}"

puts "Longest Heat Wave Length: #{longest_heat_wave(temps)} days"
puts "Longest Cold Spell Length: #{longest_cold_spell(temps)} days"
puts "Hottest Month: #{hottest_month(temps)}"
```

Output:

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming
\Lab\Ruby-Programming\Week4\Lab3_2> ruby Q1.rb
Average Temperature for the Year: 25.32°C
Hottest Temperature: 35°C on Day 19
Coldest Temperature: 15°C on Day 20
Monthly Averages: 25, 28, 24, 24, 25, 25, 26, 26, 25, 26, 25, 25
Longest Heat Wave Length: 3 days
Longest Cold Spell Length: 4 days
Hottest Month: 2
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming
\Lab\Ruby-Programming\Week4\Lab3_2>
```

Q2.

Code:

```
def find_head_number(array)

    return nil if array.length < 3

    (1...array.length - 1).each do |i|
      if array[i] > array[i - 1] && array[i] > array[i + 1]
        return array[i]
      end
    end
    nil
  end

  def find_master_pair(array)
    return nil if array.length < 2

    max_sum = 0
    best_pair = nil
```

```ruby
  (0...(array.length - 1)).each do |i|
    ((i + 1)...array.length).each do |j|
      sum = array[i] + array[j]
      if sum > max_sum
        max_sum = sum
        best_pair = [array[i], array[j]]
      end
    end
  end

  best_pair
end

array = [1, 3, 2, 8, 5, 4, 10, 12, 53, 23, 25]

head_number = find_head_number(array)
master_pair = find_master_pair(array)

puts "Head Number: #{head_number.inspect}"
puts "Master Pair: #{master_pair.inspect}"
```

Output:

```
\Lab\Ruby-Programming\Week4\Lab3_2>     ruby Q2.rb
Head Number: 3
Master Pair: [53, 25]
```

Q3.

Code:

```
class FactorialDispatcher
    def method_missing(method_name, *args)
      if method_name == :factorial
        handle_factorial(*args)
      else
        super
      end
    end

    def respond_to_missing?(method_name, include_private = false)
      method_name == :factorial || super
    end

    private
    def handle_factorial(n)
      if n.is_a?(Integer) && n >= 0
        result = factorial(n)
        puts "Result of factorial(#{n}): #{result}"
      else
        puts "Error: Factorial is only defined for non-negative integers."
      end
    end

    def factorial(n)
      (1..n).inject(:*) || 1
    end
  end

  dispatcher = FactorialDispatcher.new

  dispatcher.factorial(5)
  dispatcher.factorial(10)
  dispatcher.factorial(-1)
  dispatcher.factorial('a')
  puts dispatcher.respond_to?(:factorial)
  puts dispatcher.respond_to?(:non_existent)
```

Output

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming
\Lab\Ruby-Programming\Week4\Lab3_2> ruby Q3.rb
Result of factorial(5): 120
Result of factorial(10): 3628800
Error: Factorial is only defined for non-negative integers.
Error: Factorial is only defined for non-negative integers.
true
false
```

Q4.

Code:

```ruby
def evaluate_expression(expression)
    begin
      eval(expression)
    rescue StandardError => e
      return nil
    end
  end

  def balanced_parentheses?(str)
    stack = []
    brackets = { '(' => ')', '{' => '}', '[' => ']', '<' => '>' }
    positions = []
    expression = ""

    str.each_char.with_index do |char, index|
      if brackets.keys.include?(char)
        stack.push(char)
        positions.push(index)
      elsif brackets.values.include?(char)
        last_open = stack.pop
        if last_open.nil? || brackets[last_open] != char
          return "Mismatch at position #{index + 1}"
        end
        positions.pop
      else
        expression << char unless char.match?(/[()\[\]{}<>]/)
      end
    end
```

```ruby
  if stack.empty?
    result = evaluate_expression(expression.strip)
    return result.nil? ? true : result
  else
    return "Mismatch at position #{positions.last + 1}"
  end
end

puts balanced_parentheses?("(1+2)*{3+4}")
puts balanced_parentheses?("a ( b c ) d")
puts balanced_parentheses?("{[<>]}")
puts balanced_parentheses?("{[>]}")
```

Output:

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming
\Lab\Ruby-Programming\Week4\Lab3_2> ruby Q4.rb
11
true
true
Mismatch at position 3
```

Q5.

TextFile:

song_lyrics.txt

```
Don't want to be a fool for you
Just another player in your game for two
You may hate me but it ain't no lie
Baby bye bye bye
Bye bye
```

Code:

```
def word_frequency(file_path)
    word_count = Hash.new(0)
    text = File.read(file_path)
    words = text.downcase.scan(/\b[\w']+\b/)
    words.each { |word| word_count[word] += 1 }
    word_count.each { |word, count| puts "#{word}: #{count}" }
    most_frequent_word = word_count.max_by { |_, count| count }
    puts "\nMost frequently used word: '#{most_frequent_word[0]}' appears
#{most_frequent_word[1]} times"
  end

  word_frequency('song_lyrics.txt')
```

Output:

```
PS D:\VIT\Academics\Fall Semester 24-25\SWE2034 - Ruby Programming
\Lab\Ruby-Programming\Week4\Lab3_2> ruby Q5.rb
don't: 1
want: 1
to: 1
be: 1
a: 1
fool: 1
for: 2
you: 2
just: 1
another: 1
player: 1
in: 1
your: 1
game: 1
two: 1
may: 1
hate: 1
me: 1
but: 1
it: 1
ain't: 1
no: 1
lie: 1
baby: 1
bye: 5

Most frequently used word: 'bye' appears 5 times
```