



**VIT<sup>®</sup>**  
**CHENNAI**  
Vellore Institute of Technology  
(Deemed to be University under section 3 of UGC Act, 1956)

---

## **SWE2034 – Ruby Programming**

**Guided By – Dr Yogesh C**

**Slot – L5+L6**

---

**NAME: VISHWANTH P**

**REGISTER.NO: 21MIS1117**

---

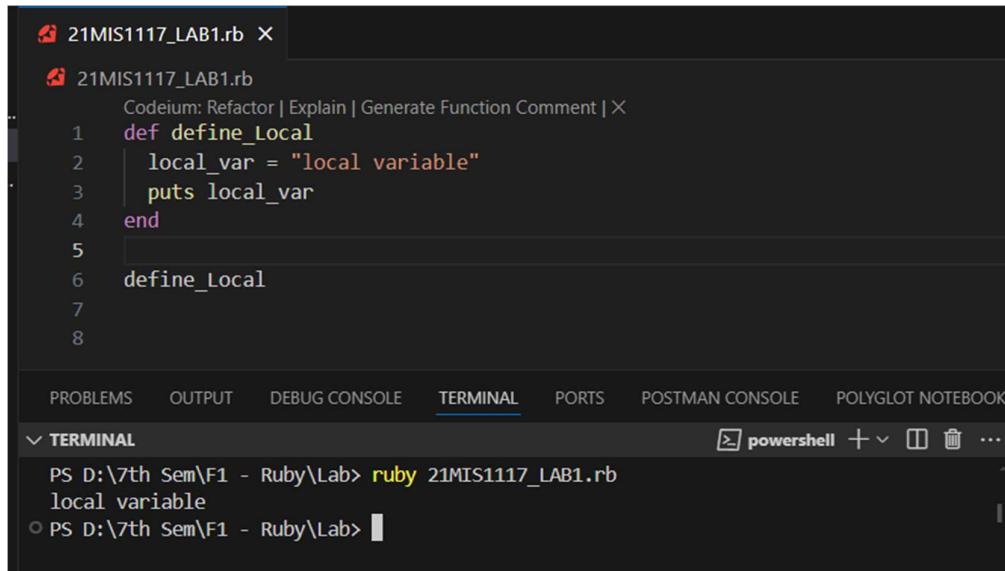
**Lab Assessment - 1**

## 1. Introduction

### a. Variables and Methods

#### i. Different Types of Variables and Scope

### Local Variable



The screenshot shows a VS Code editor with a file named 21MIS1117\_LAB1.rb. The code defines a method `define_local` that creates a local variable `local_var` and prints its value. The terminal shows the command `ruby 21MIS1117_LAB1.rb` being executed, resulting in the output `local variable`.

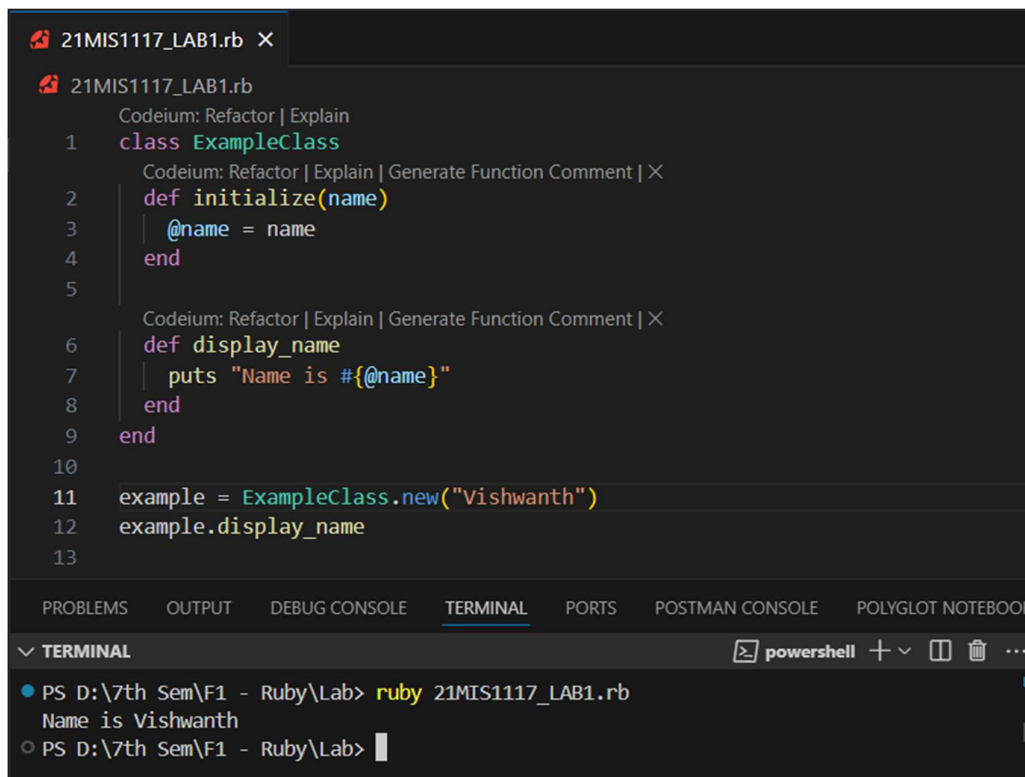
```
21MIS1117_LAB1.rb
Codeium: Refactor | Explain | Generate Function Comment | X
1 def define_local
2   local_var = "local variable"
3   puts local_var
4 end
5
6 define_local
7
8
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK

▼ **TERMINAL** powershell + - [ ] [X] ...

PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
local variable  
PS D:\7th Sem\F1 - Ruby\Lab>

### Instance Variable



The screenshot shows a VS Code editor with a file named 21MIS1117\_LAB1.rb. The code defines a class `ExampleClass` with an instance variable `@name` and a method `display_name` that prints the value of `@name`. The terminal shows the command `ruby 21MIS1117_LAB1.rb` being executed, resulting in the output `Name is Vishwanth`.

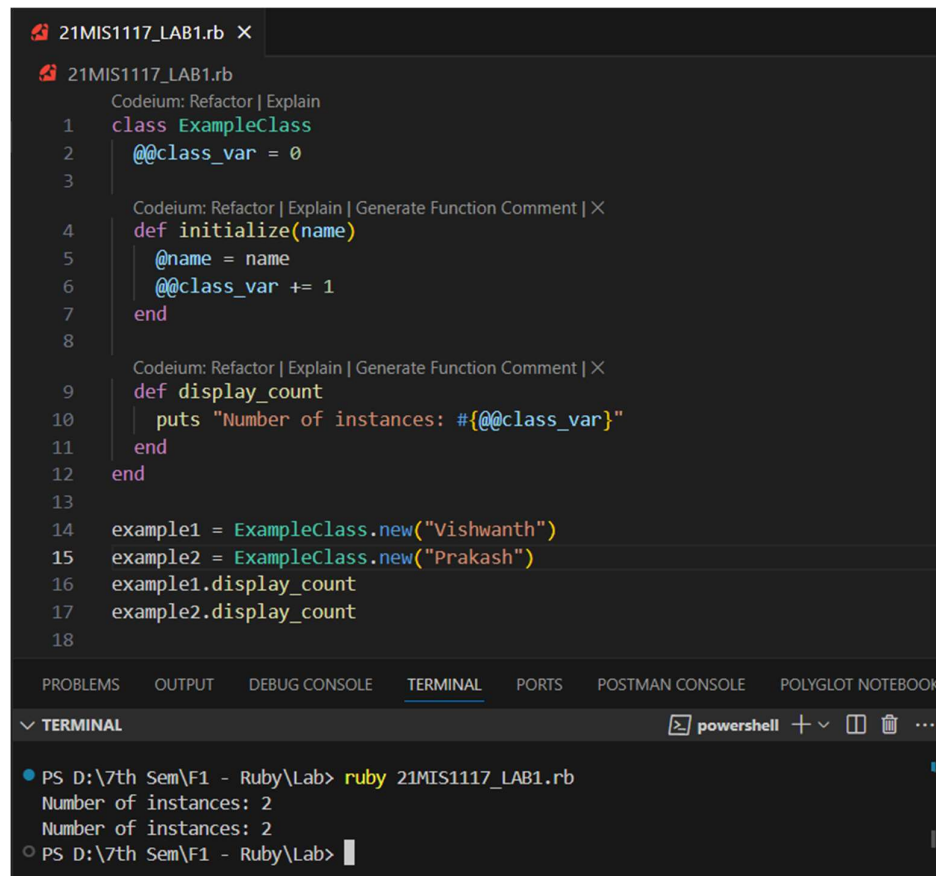
```
21MIS1117_LAB1.rb
Codeium: Refactor | Explain
1 class ExampleClass
2   Codeium: Refactor | Explain | Generate Function Comment | X
3   def initialize(name)
4     @name = name
5   end
6
7   Codeium: Refactor | Explain | Generate Function Comment | X
8   def display_name
9     puts "Name is #{@name}"
10  end
11 end
12
13 example = ExampleClass.new("Vishwanth")
14 example.display_name
15
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK

▼ **TERMINAL** powershell + - [ ] [X] ...

● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
Name is Vishwanth  
○ PS D:\7th Sem\F1 - Ruby\Lab>

## Class Variable



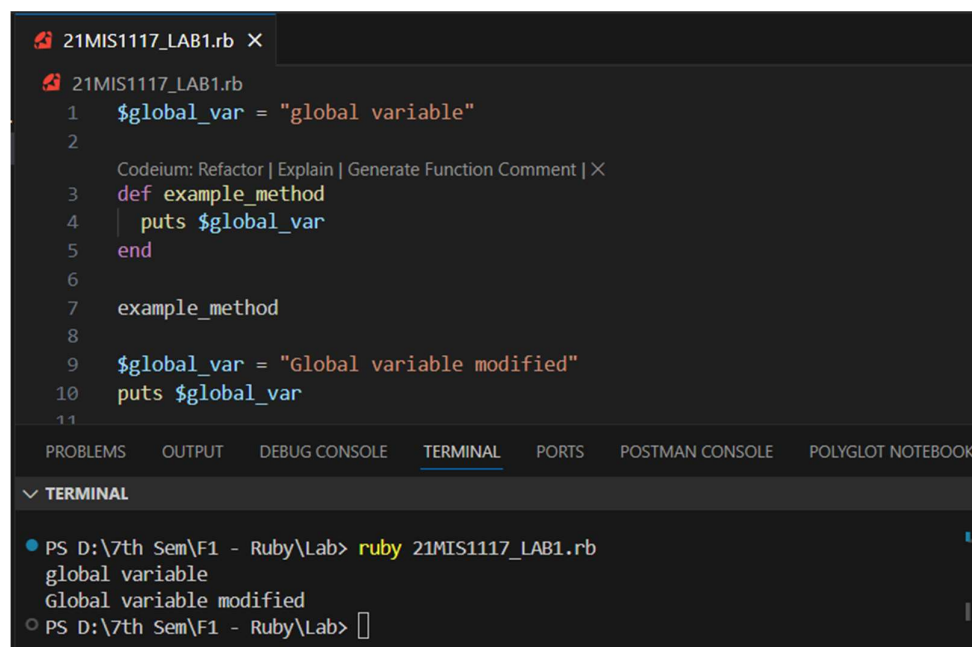
The screenshot shows a Ruby IDE with a file named 21MIS1117\_LAB1.rb. The code defines a class `ExampleClass` with a class variable `@@class_var` initialized to 0. The `initialize` method takes a name and increments the class variable. The `display_count` method prints the current value of the class variable. Two instances are created: `example1` with name "Vishwanth" and `example2` with name "Prakash". Both instances call `display_count`, which prints "Number of instances: 2" twice.

```
1 class ExampleClass
2   @@class_var = 0
3
4   def initialize(name)
5     @name = name
6     @@class_var += 1
7   end
8
9   def display_count
10    puts "Number of instances: #{@@class_var}"
11  end
12 end
13
14 example1 = ExampleClass.new("Vishwanth")
15 example2 = ExampleClass.new("Prakash")
16 example1.display_count
17 example2.display_count
18
```

TERMINAL

```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Number of instances: 2
Number of instances: 2
PS D:\7th Sem\F1 - Ruby\Lab>
```

## Global Variables



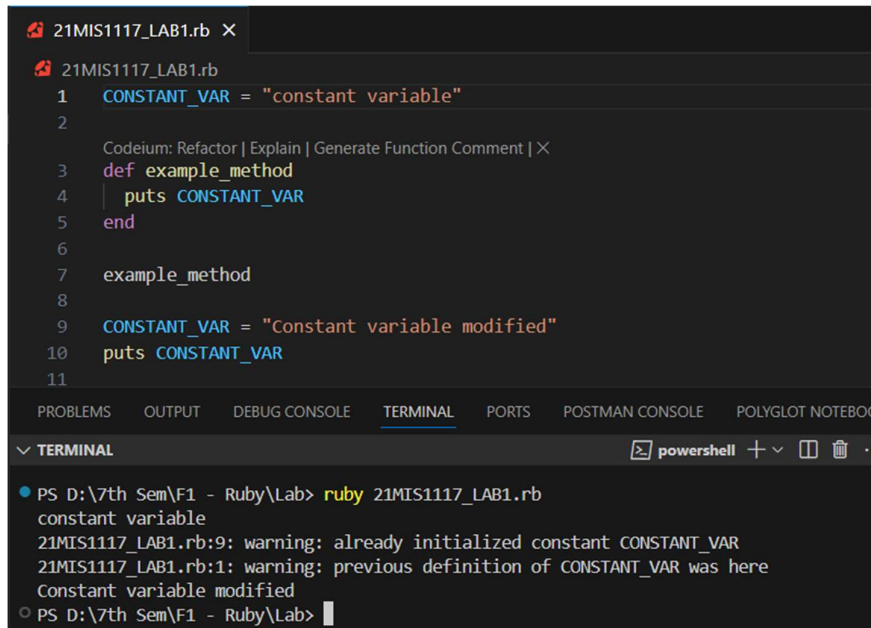
The screenshot shows a Ruby IDE with a file named 21MIS1117\_LAB1.rb. The code defines a global variable `$global_var` with the value "global variable". A method `example_method` is defined that prints the value of `$global_var`. The method is called, and then the global variable is modified to "Global variable modified" and printed again.

```
1 $global_var = "global variable"
2
3 def example_method
4   puts $global_var
5 end
6
7 example_method
8
9 $global_var = "Global variable modified"
10 puts $global_var
11
```

TERMINAL

```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
global variable
Global variable modified
PS D:\7th Sem\F1 - Ruby\Lab>
```

## Constants



```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  CONSTANT_VAR = "constant variable"
2
3  Codeium: Refactor | Explain | Generate Function Comment | X
4  def example_method
5    puts CONSTANT_VAR
6  end
7
8  example_method
9
10 CONSTANT_VAR = "Constant variable modified"
11 puts CONSTANT_VAR
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK

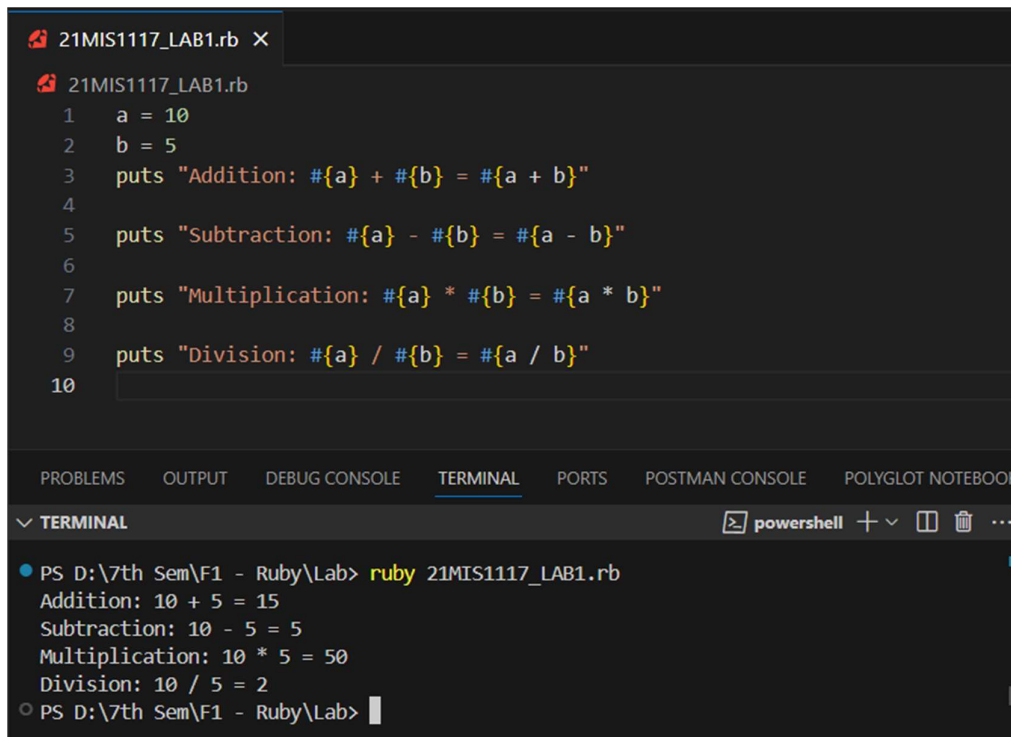
▼ **TERMINAL** powershell + -

```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
constant variable
21MIS1117_LAB1.rb:9: warning: already initialized constant CONSTANT_VAR
21MIS1117_LAB1.rb:1: warning: previous definition of CONSTANT_VAR was here
Constant variable modified
PS D:\7th Sem\F1 - Ruby\Lab>
```

## ii. Arithmetic, Relation, Logical and Boolean operators (Each 2 Programs)

### Arithmetic Operators

#### Program 1: Basic Arithmetic Operations



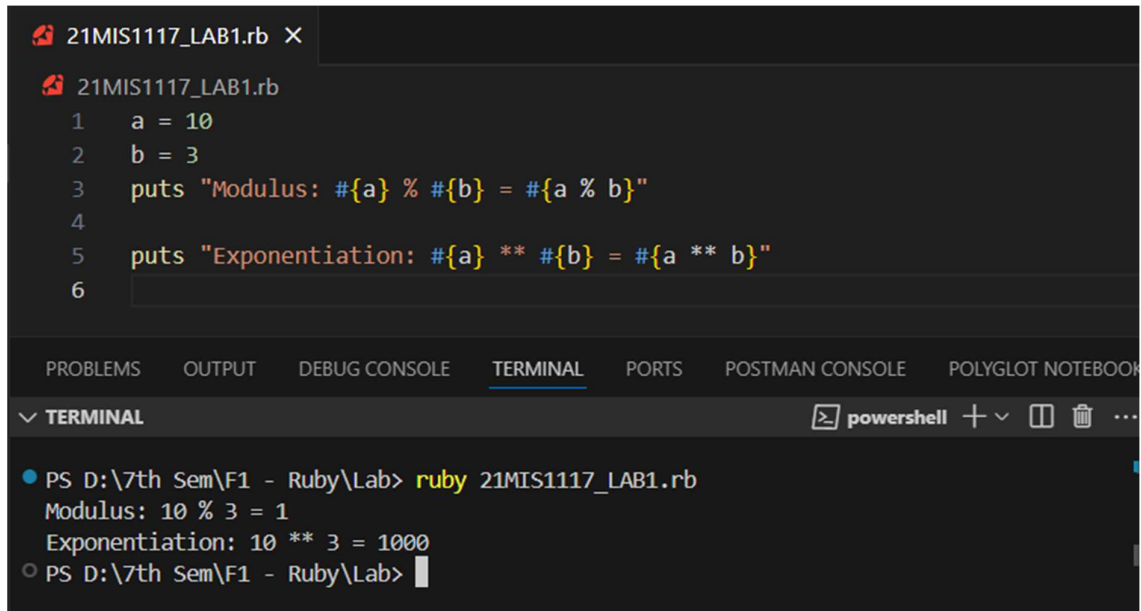
```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  a = 10
2  b = 5
3  puts "Addition: #{a} + #{b} = #{a + b}"
4
5  puts "Subtraction: #{a} - #{b} = #{a - b}"
6
7  puts "Multiplication: #{a} * #{b} = #{a * b}"
8
9  puts "Division: #{a} / #{b} = #{a / b}"
10
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK

▼ **TERMINAL** powershell + -

```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Addition: 10 + 5 = 15
Subtraction: 10 - 5 = 5
Multiplication: 10 * 5 = 50
Division: 10 / 5 = 2
PS D:\7th Sem\F1 - Ruby\Lab>
```

## Program 2: Modulus and Exponentiation



The screenshot shows an IDE with a file named 21MIS1117\_LAB1.rb. The code defines variables a = 10 and b = 3, then prints the modulus and exponentiation results. The terminal output shows the program execution results.

```
21MIS1117_LAB1.rb
1  a = 10
2  b = 3
3  puts "Modulus: #{a} % #{b} = #{a % b}"
4
5  puts "Exponentiation: #{a} ** #{b} = #{a ** b}"
6

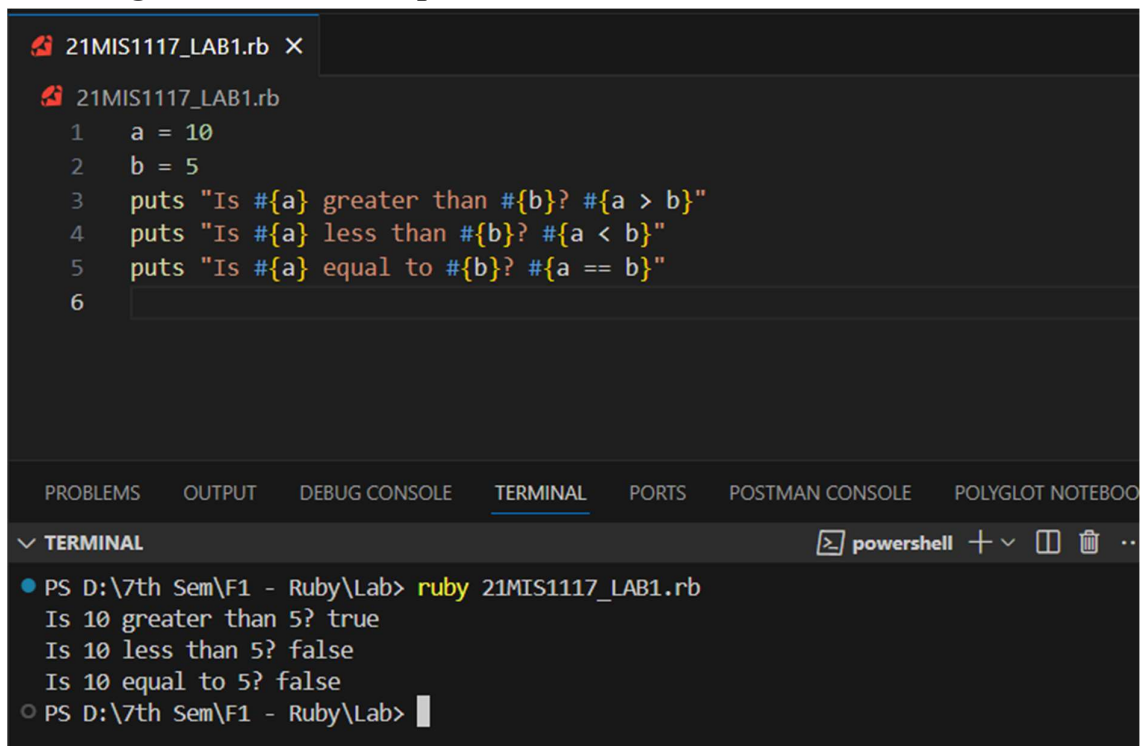
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  POLYGLOT NOTEBOOK

▼ TERMINAL powershell + - □ ✕ ...

● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Modulus: 10 % 3 = 1
Exponentiation: 10 ** 3 = 1000
○ PS D:\7th Sem\F1 - Ruby\Lab> 
```

## Relational Operators

### Program 1: Basic Comparisons



The screenshot shows an IDE with a file named 21MIS1117\_LAB1.rb. The code defines variables a = 10 and b = 5, then prints the results of three relational comparisons: greater than, less than, and equal to. The terminal output shows the program execution results.

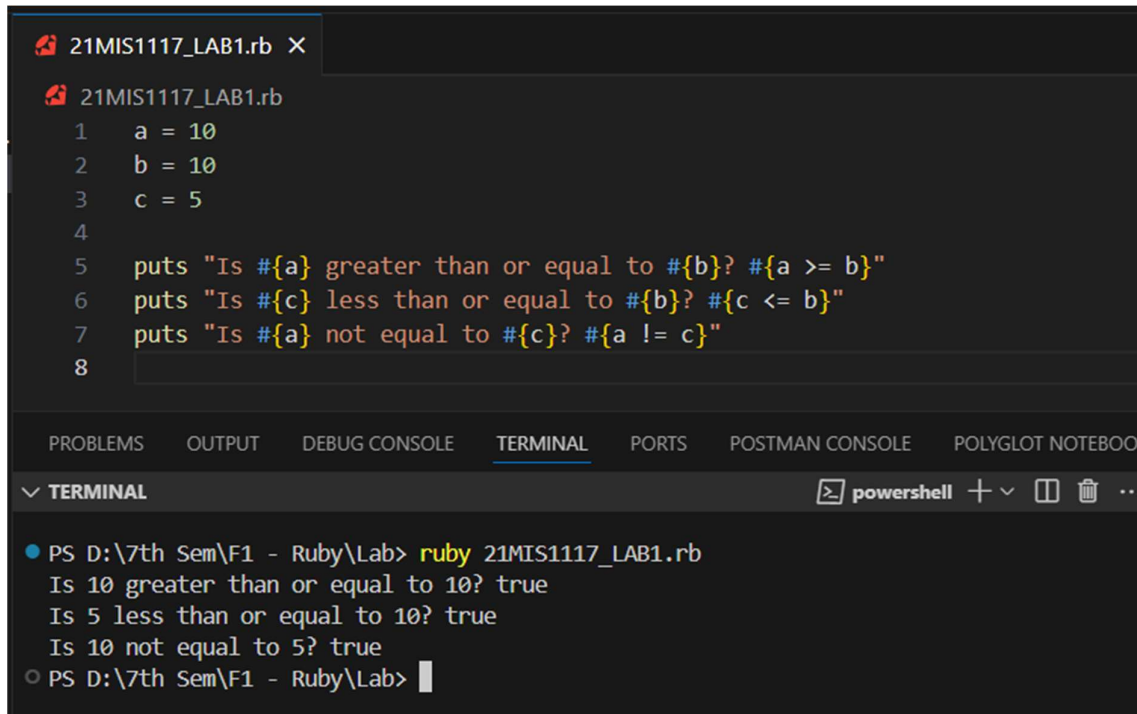
```
21MIS1117_LAB1.rb
1  a = 10
2  b = 5
3  puts "Is #{a} greater than #{b}? #{a > b}"
4  puts "Is #{a} less than #{b}? #{a < b}"
5  puts "Is #{a} equal to #{b}? #{a == b}"
6

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  POLYGLOT NOTEBOOK

▼ TERMINAL powershell + - □ ✕ ..

● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Is 10 greater than 5? true
Is 10 less than 5? false
Is 10 equal to 5? false
○ PS D:\7th Sem\F1 - Ruby\Lab> 
```

## Program 2: Complex Comparisons

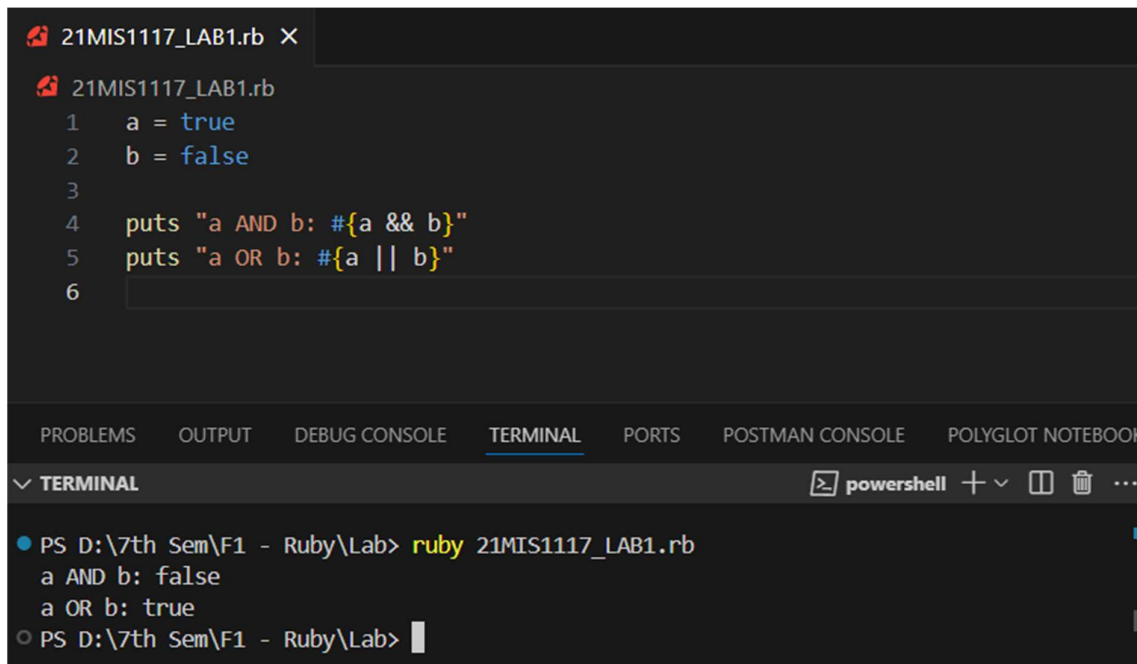


```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  a = 10
2  b = 10
3  c = 5
4
5  puts "Is #{a} greater than or equal to #{b}? #{a >= b}"
6  puts "Is #{c} less than or equal to #{b}? #{c <= b}"
7  puts "Is #{a} not equal to #{c}? #{a != c}"
8

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK
▼ TERMINAL powershell + - [] ...
● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Is 10 greater than or equal to 10? true
Is 5 less than or equal to 10? true
Is 10 not equal to 5? true
○ PS D:\7th Sem\F1 - Ruby\Lab> |
```

## Logical Operators

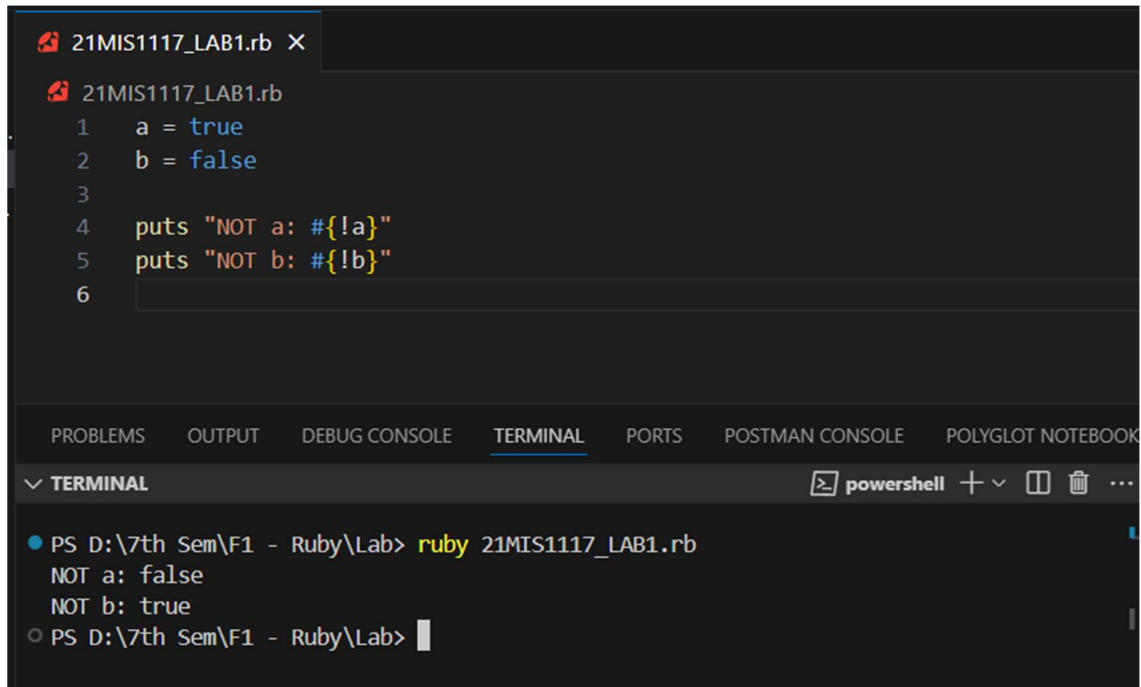
### Program 1: Logical AND and OR



```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  a = true
2  b = false
3
4  puts "a AND b: #{a && b}"
5  puts "a OR b: #{a || b}"
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK
▼ TERMINAL powershell + - [] ...
● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
a AND b: false
a OR b: true
○ PS D:\7th Sem\F1 - Ruby\Lab> |
```

## Program 2: Logical NOT



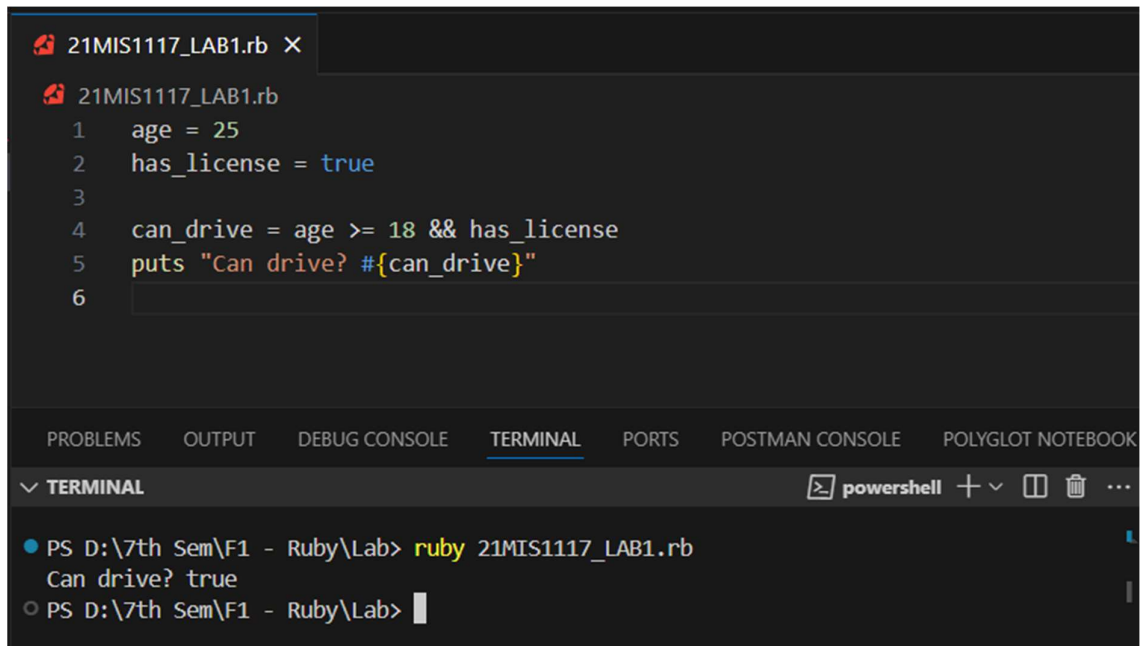
The screenshot shows an IDE with a file named 21MIS1117\_LAB1.rb. The code defines two variables, `a` and `b`, and prints their logical NOT values. The terminal output shows that `NOT a` is `false` and `NOT b` is `true`.

```
21MIS1117_LAB1.rb
1  a = true
2  b = false
3
4  puts "NOT a: #{!a}"
5  puts "NOT b: #{!b}"
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK
v TERMINAL powershell + - [ ] [ ] ...
• PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
  NOT a: false
  NOT b: true
○ PS D:\7th Sem\F1 - Ruby\Lab> 
```

## Boolean Operators

### Program 1: Combining Conditions



The screenshot shows an IDE with a file named 21MIS1117\_LAB1.rb. The code defines variables for age and license status, then checks if a person can drive based on these conditions. The terminal output shows that the person can drive.

```
21MIS1117_LAB1.rb
1  age = 25
2  has_license = true
3
4  can_drive = age >= 18 && has_license
5  puts "Can drive? #{can_drive}"
6

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK
v TERMINAL powershell + - [ ] [ ] ...
• PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
  Can drive? true
○ PS D:\7th Sem\F1 - Ruby\Lab> 
```



## Program 2: Using OR to Provide Alternatives

```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  is_weekend = true
2  is_holiday = false
3
4  can_relax = is_weekend || is_holiday
5  puts "Can relax? #{can_relax}"
6

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  POLYGLOT NOTEBOOK
▼ TERMINAL  powershell + - [ ] [X] ...
● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Can relax? true
○ PS D:\7th Sem\F1 - Ruby\Lab> 
```

### b. Conversion of Variables

```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  str_num = "123"
2  str_float = "123.45"
3  num = 123
4  array = [[:key1, "value1"], [:key2, "value2"]]
5  hash = {key1: "value1", key2: "value2"}
6
7  int_val = str_num.to_i
8  puts "String '#{str_num}' to Integer: #{int_val} (#{int_val.class})"
9
10 float_val = str_float.to_f
11 puts "String '#{str_float}' to Float: #{float_val} (#{float_val.class})"
12
13 str_from_int = num.to_s
14 puts "Integer #{num} to String: '#{str_from_int}' (#{str_from_int.class})"
15
16 symbol = str_num.to_sym
17 puts "String '#{str_num}' to Symbol: :#{symbol} (#{symbol.class})"
18
19 hash_from_array = array.to_h
20 puts "Array #{array} to Hash: #{hash_from_array} (#{hash_from_array.class})"
21
22 array_from_hash = hash.to_a
23 puts "Hash #{hash} to Array: #{array_from_hash} (#{array_from_hash.class})"
24
25 array = [1, 2, 3]
26 str_from_array = array.join(", ")
27 new_array = str_from_array.split(", ").map(&:to_i)
28 puts "Array #{array} to String '#{str_from_array}' and back to Array #{new_array} (#{new_array.class})"
29
```



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK SERIAL MONITOR
▼ TERMINAL powershell + - [] ...
● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
String '123' to Integer: 123 (Integer)
String '123.45' to Float: 123.45 (Float)
Integer 123 to String: '123' (String)
String '123' to Symbol: :123 (Symbol)
Array [[:key1, "value1"], [:key2, "value2"]] to Hash: {:key1=>"value1", :key2=>"value2"} (Hash)
Hash {:key1=>"value1", :key2=>"value2"} to Array: [[:key1, "value1"], [:key2, "value2"]] (Array)
Array [1, 2, 3] to String '1, 2, 3' and back to Array [1, 2, 3] (Array)
○ PS D:\7th Sem\F1 - Ruby\Lab>
```

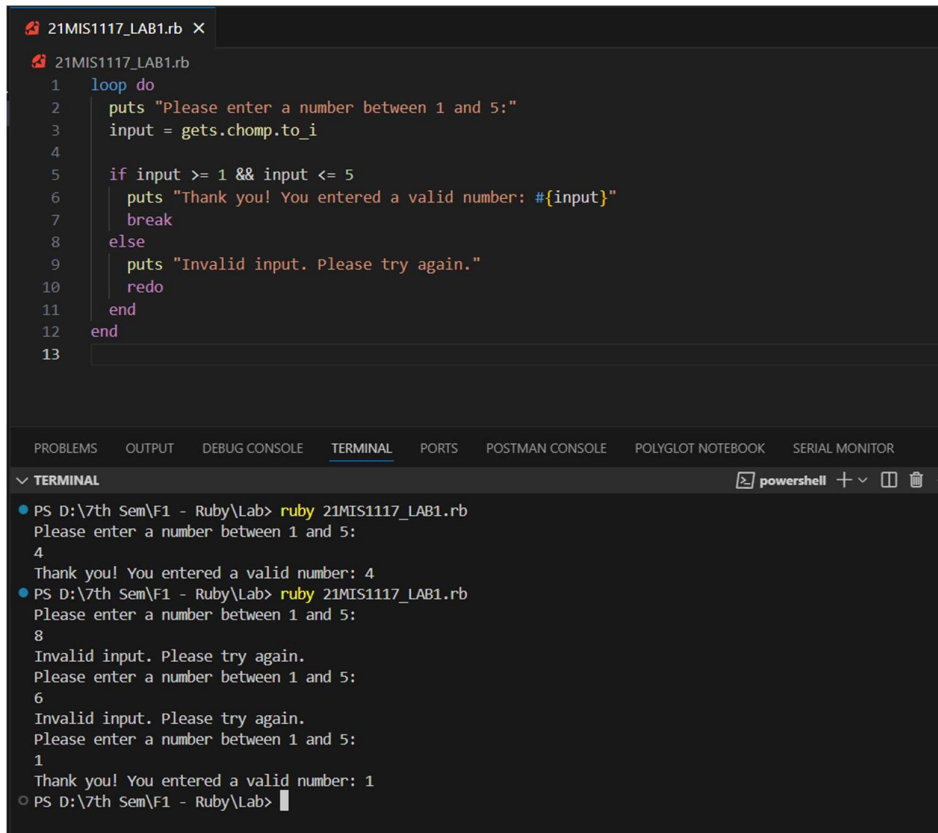
### c. Inclusive and Non-Inclusive Ranges

```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1 inclusive_range = 1..5
2 puts "Inclusive Range (1..5):"
3 inclusive_range.each do |num|
4   print "#{num} "
5 end
6 puts "\nDoes the inclusive range include 5? #{inclusive_range.include?(5)}"
7
8 non_inclusive_range = 1...5
9 puts "\nNon-Inclusive Range (1...5):"
10 non_inclusive_range.each do |num|
11   print "#{num} "
12 end
13 puts "\nDoes the non-inclusive range include 5? #{non_inclusive_range.include?(5)}"
14

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK SERIAL MONITOR
▼ TERMINAL powershell + - [] ...
● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Inclusive Range (1..5):
1 2 3 4 5
Does the inclusive range include 5? true

Non-Inclusive Range (1...5):
1 2 3 4
Does the non-inclusive range include 5? false
○ PS D:\7th Sem\F1 - Ruby\Lab>
```

## d. Redo Program

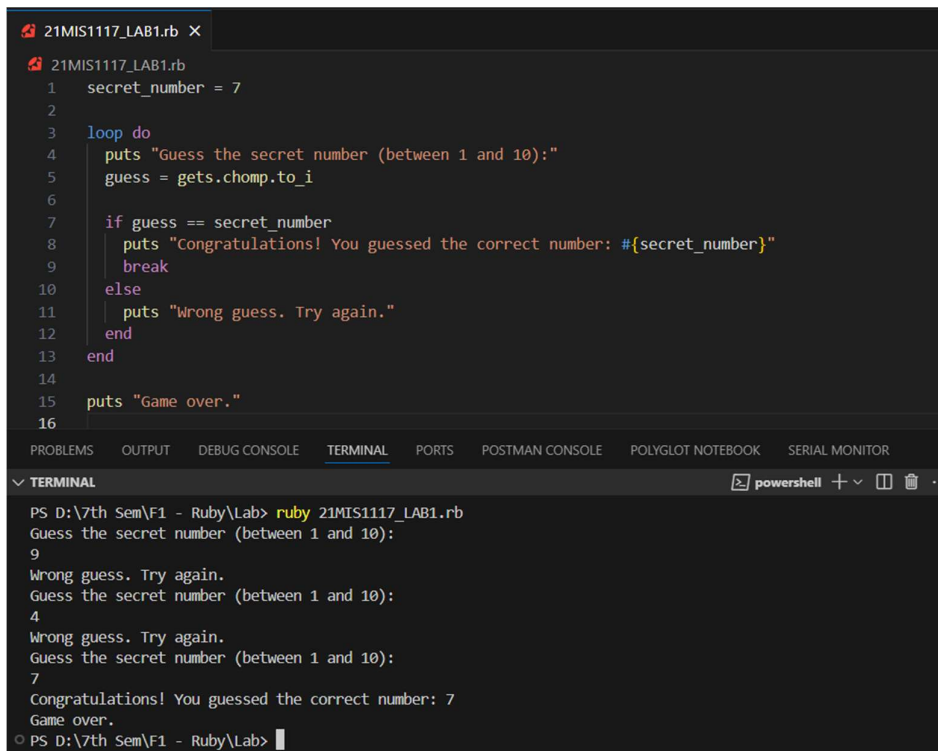


```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  loop do
2    puts "Please enter a number between 1 and 5:"
3    input = gets.chomp.to_i
4
5    if input >= 1 && input <= 5
6      puts "Thank you! You entered a valid number: #{input}"
7      break
8    else
9      puts "Invalid input. Please try again."
10     redo
11   end
12 end
13
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK SERIAL MONITOR

PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
Please enter a number between 1 and 5:  
4  
Thank you! You entered a valid number: 4  
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
Please enter a number between 1 and 5:  
8  
Invalid input. Please try again.  
Please enter a number between 1 and 5:  
6  
Invalid input. Please try again.  
Please enter a number between 1 and 5:  
1  
Thank you! You entered a valid number: 1  
PS D:\7th Sem\F1 - Ruby\Lab>

## e. Break Program

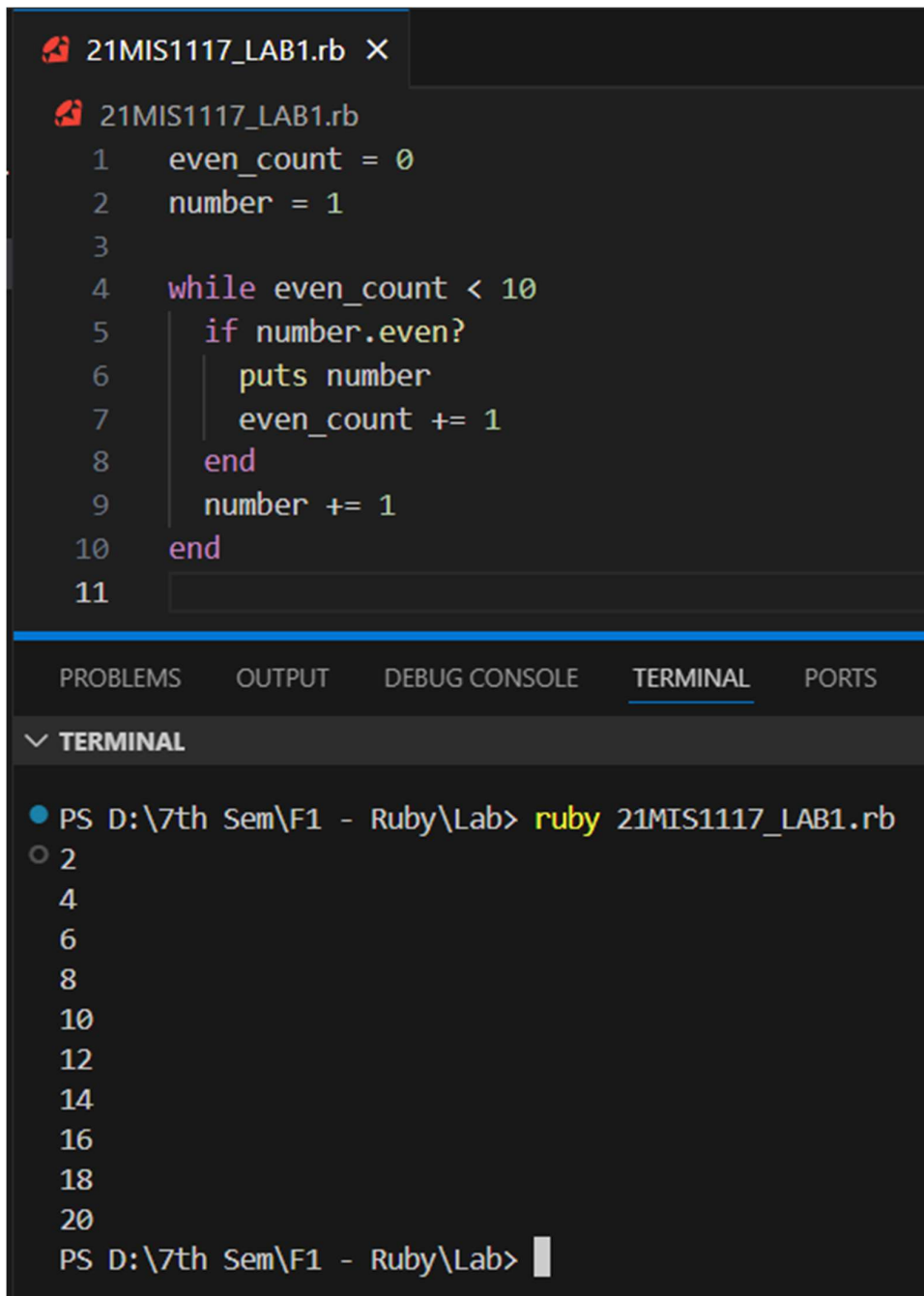


```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  secret_number = 7
2
3  loop do
4    puts "Guess the secret number (between 1 and 10):"
5    guess = gets.chomp.to_i
6
7    if guess == secret_number
8      puts "Congratulations! You guessed the correct number: #{secret_number}"
9      break
10   else
11     puts "Wrong guess. Try again."
12   end
13 end
14
15 puts "Game over."
16
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGLOT NOTEBOOK SERIAL MONITOR

PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
Guess the secret number (between 1 and 10):  
9  
Wrong guess. Try again.  
Guess the secret number (between 1 and 10):  
4  
Wrong guess. Try again.  
Guess the secret number (between 1 and 10):  
7  
Congratulations! You guessed the correct number: 7  
Game over.  
PS D:\7th Sem\F1 - Ruby\Lab>

## 2. A program that prints out the first 10 even numbers



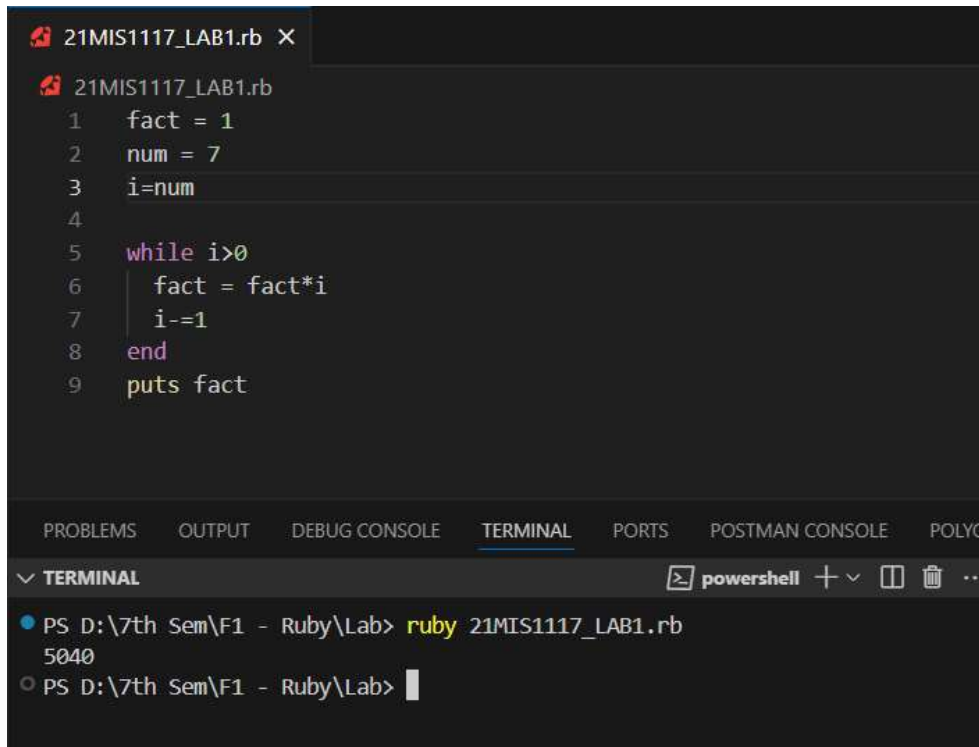
The screenshot shows a Ruby IDE with a file named `21MIS1117_LAB1.rb`. The code in the editor is as follows:

```
1  even_count = 0
2  number = 1
3
4  while even_count < 10
5    if number.even?
6      puts number
7      even_count += 1
8    end
9    number += 1
10 end
11
```

Below the editor, the **TERMINAL** tab is active, showing the command `ruby 21MIS1117_LAB1.rb` and its output:

```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
2
4
6
8
10
12
14
16
18
20
PS D:\7th Sem\F1 - Ruby\Lab>
```

### 3. A program that calculates the factorial of a number using a while loop



```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  fact = 1
2  num = 7
3  i=num
4
5  while i>0
6    fact = fact*i
7    i-=1
8  end
9  puts fact
```

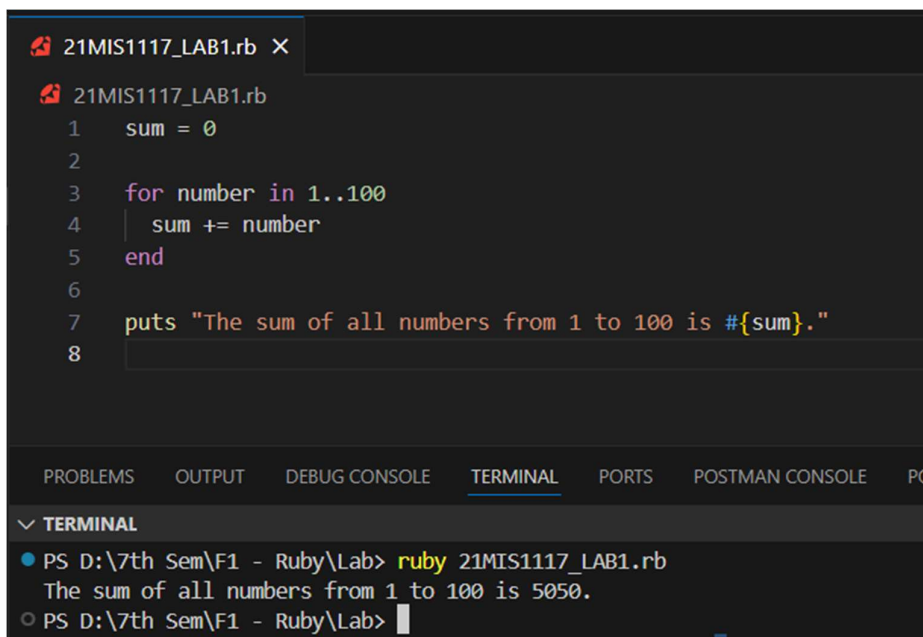
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGL

▼ **TERMINAL** powershell + -

● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
5040

○ PS D:\7th Sem\F1 - Ruby\Lab>

### 4. A program that prints out the sum of all numbers from 1 to 100 using a for loop S.



```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  sum = 0
2
3  for number in 1..100
4    sum += number
5  end
6
7  puts "The sum of all numbers from 1 to 100 is #{sum}."
8
```

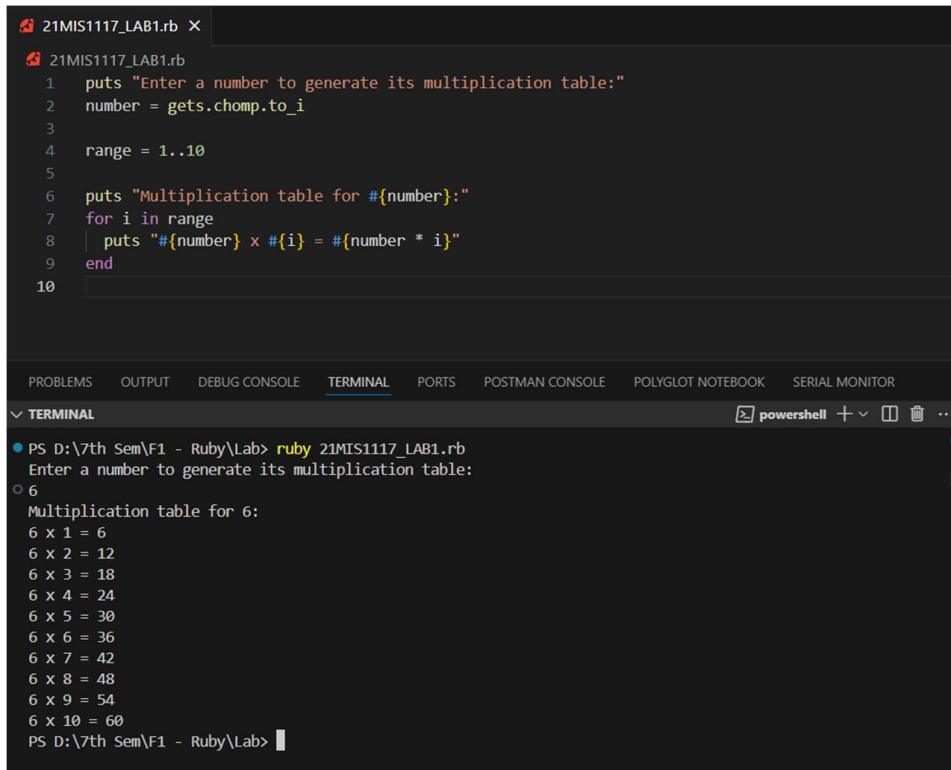
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE PO

▼ **TERMINAL**

● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
The sum of all numbers from 1 to 100 is 5050.

○ PS D:\7th Sem\F1 - Ruby\Lab>

## 5. A program that prints out the multiplication table of a number



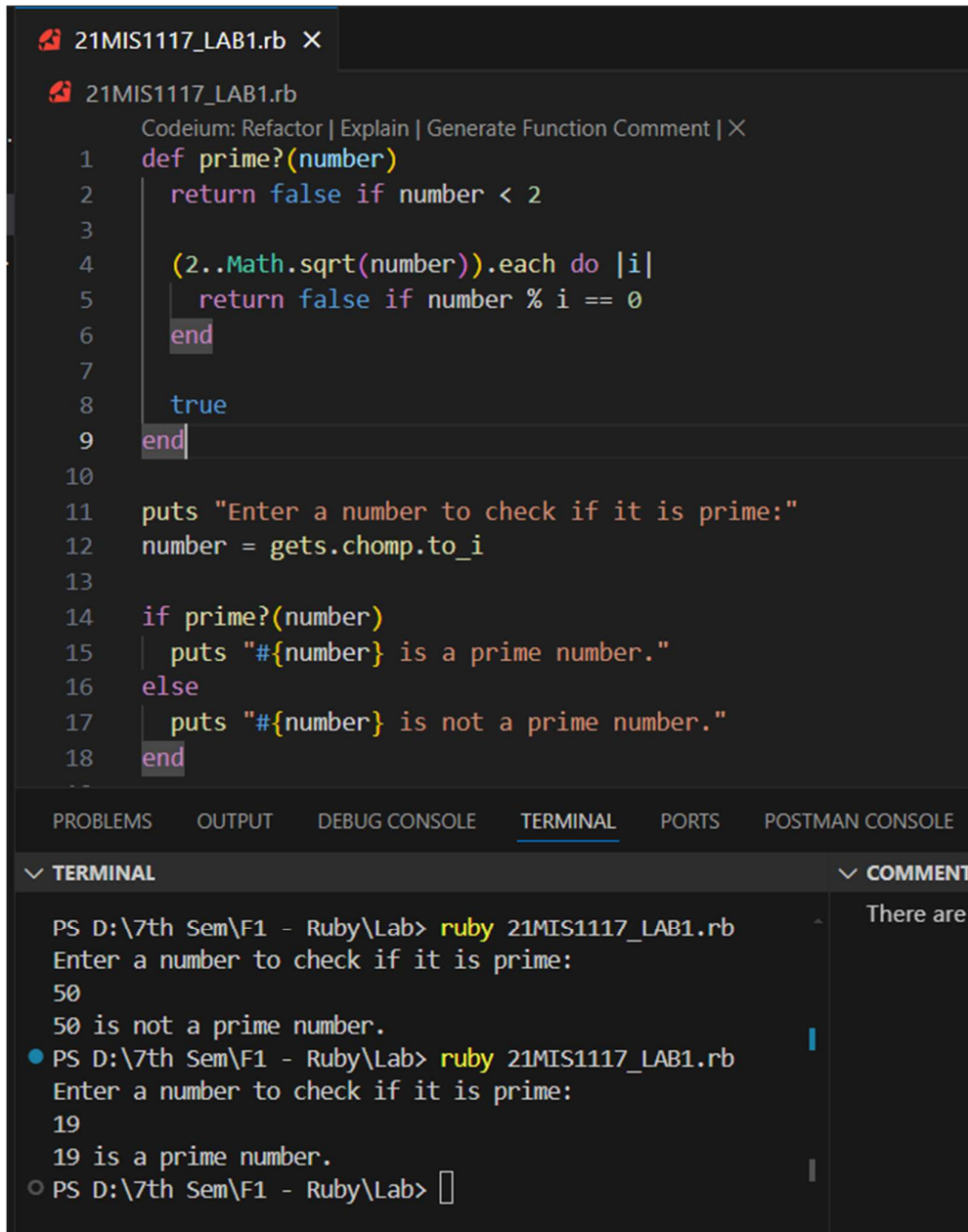
The screenshot shows a Ruby IDE with a file named `21MIS1117_LAB1.rb`. The code in the editor is as follows:

```
1 puts "Enter a number to generate its multiplication table:"
2 number = gets.chomp.to_i
3
4 range = 1..10
5
6 puts "Multiplication table for #{number}:"
7 for i in range
8   puts "#{number} x #{i} = #{number * i}"
9 end
10
```

Below the editor, the **TERMINAL** tab is active, showing the execution of the program. The user has entered the number 6, and the program has printed the multiplication table for 6.

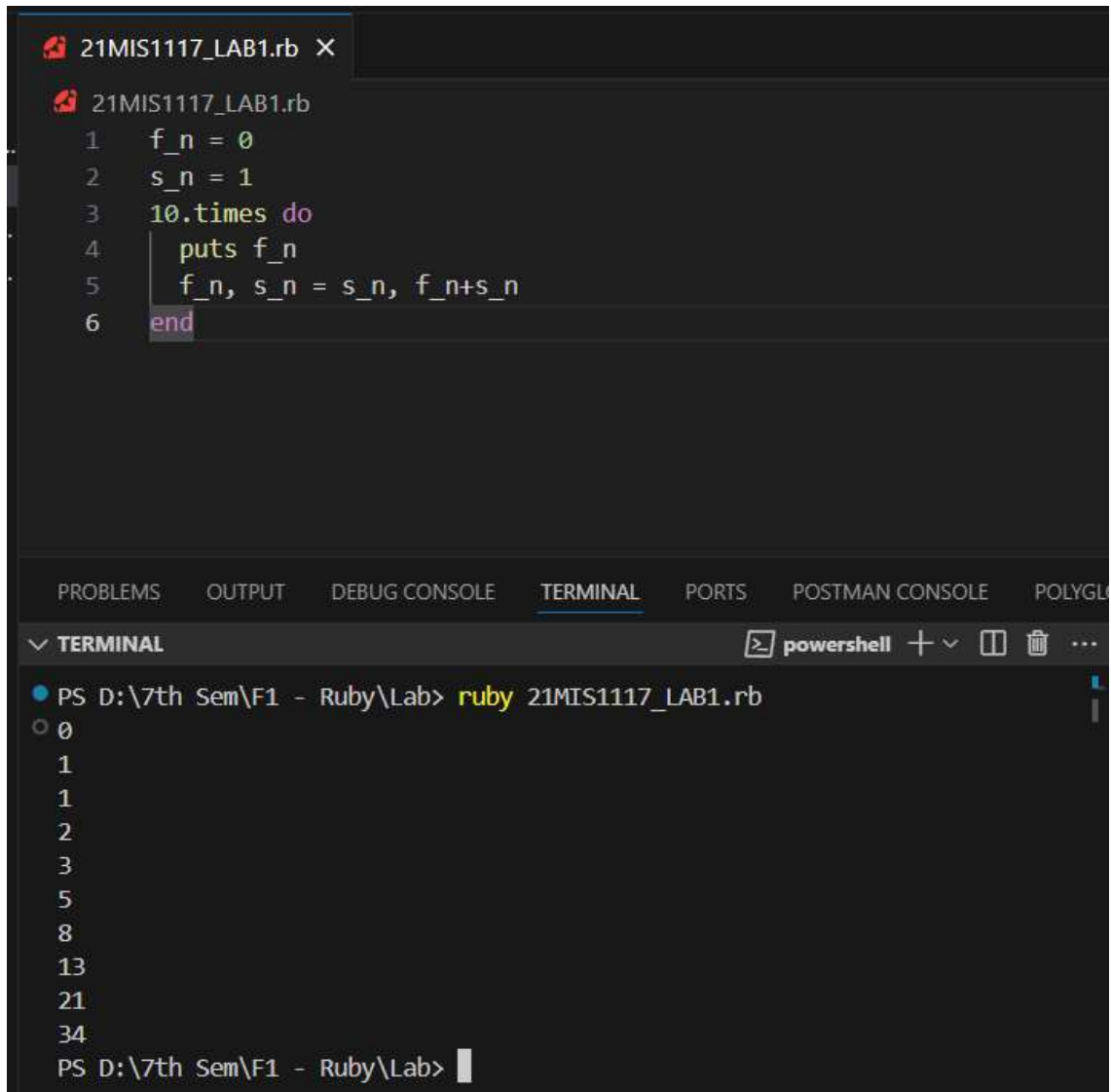
```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Enter a number to generate its multiplication table:
6
Multiplication table for 6:
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
PS D:\7th Sem\F1 - Ruby\Lab>
```

## 6. A program that checks if a number is prime or not



```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
Codeium: Refactor | Explain | Generate Function Comment | X
1 def prime?(number)
2   return false if number < 2
3
4   (2..Math.sqrt(number)).each do |i|
5     return false if number % i == 0
6   end
7
8   true
9 end
10
11 puts "Enter a number to check if it is prime:"
12 number = gets.chomp.to_i
13
14 if prime?(number)
15   puts "#{number} is a prime number."
16 else
17   puts "#{number} is not a prime number."
18 end
19
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
▼ TERMINAL ▼ COMMENT
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Enter a number to check if it is prime:
50
50 is not a prime number.
● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Enter a number to check if it is prime:
19
19 is a prime number.
○ PS D:\7th Sem\F1 - Ruby\Lab> 
```

**7. A program that prints out the Fibonacci sequence up to a certain number**



The screenshot shows a Ruby IDE with a file named `21MIS1117_LAB1.rb`. The code defines two variables, `f_n` and `s_n`, and uses a `10.times` loop to print the Fibonacci sequence. The output in the terminal shows the sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  f_n = 0
2  s_n = 1
3  10.times do
4    puts f_n
5    f_n, s_n = s_n, f_n+s_n
6  end
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS POSTMAN CONSOLE POLYGL

▼ **TERMINAL** powershell + ▢ ▢ ...

● PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb

○ 0

1

1

2

3

5

8

13

21

34

PS D:\7th Sem\F1 - Ruby\Lab> █



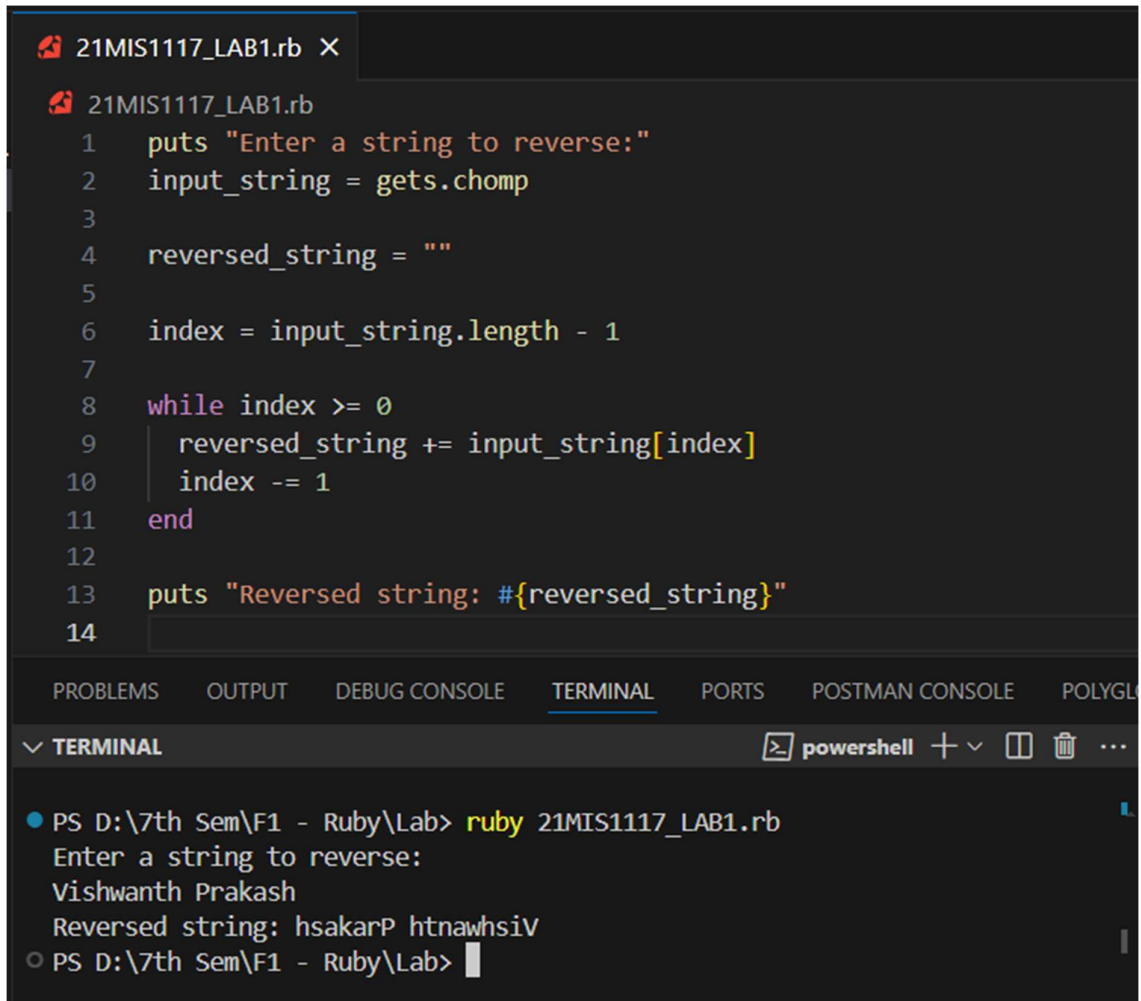
## 8. A program that prints out a triangle of stars

```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1 puts "Enter the number of rows for the triangle:"
2 rows = gets.chomp.to_i
3
4 (1..rows).each do |i|
5   puts "*" * i
6 end
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLOT
▼ TERMINAL

PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Enter the number of rows for the triangle:
9
*
**
***
****
*****
*****
*****
*****
*****
*****
*****
PS D:\7th Sem\F1 - Ruby\Lab> 
```

## 9. A program that reverses a string using a while loop

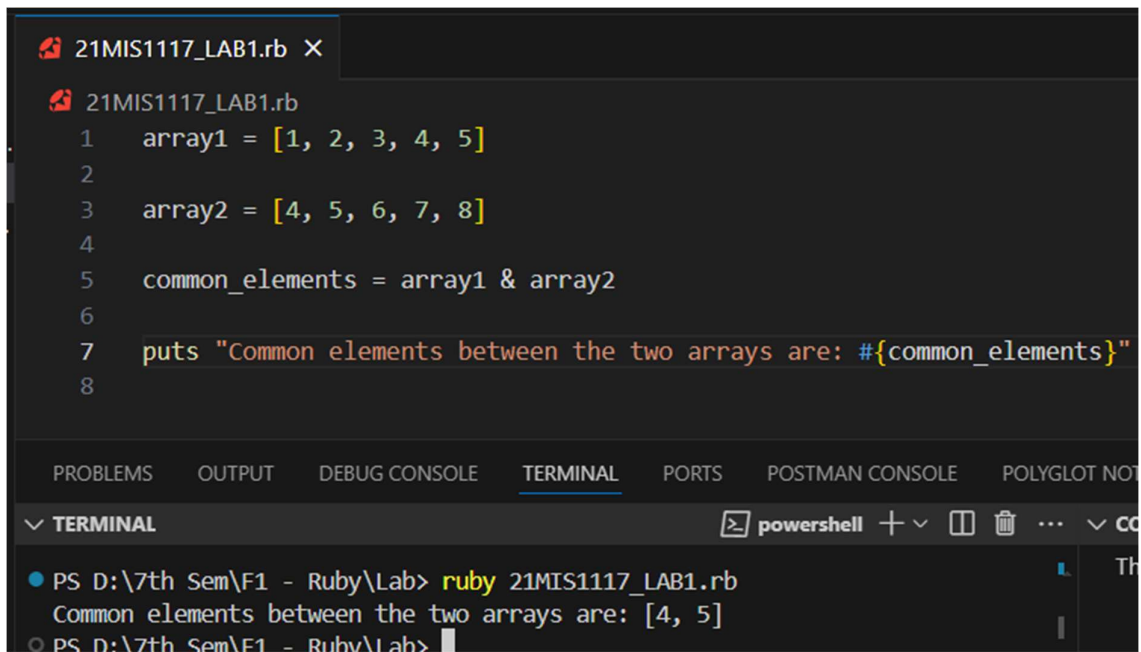


```
21MIS1117_LAB1.rb X
21MIS1117_LAB1.rb
1  puts "Enter a string to reverse:"
2  input_string = gets.chomp
3
4  reversed_string = ""
5
6  index = input_string.length - 1
7
8  while index >= 0
9      reversed_string += input_string[index]
10     index -= 1
11 end
12
13 puts "Reversed string: #{reversed_string}"
14
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGL

✓ **TERMINAL** powershell + ▾ □ 🗑 ...

- PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117\_LAB1.rb  
Enter a string to reverse:  
Vishwanth Prakash  
Reversed string: hsakarP htnawhsiV
- PS D:\7th Sem\F1 - Ruby\Lab>

**10. A program that prints out the comm-elements between two arrays**

The screenshot shows a code editor window titled '21MIS1117\_LAB1.rb' with the following Ruby code:

```
1 array1 = [1, 2, 3, 4, 5]
2
3 array2 = [4, 5, 6, 7, 8]
4
5 common_elements = array1 & array2
6
7 puts "Common elements between the two arrays are: #{common_elements}"
8
```

Below the code editor is a terminal window titled 'TERMINAL' with a 'powershell' icon. It shows the command to run the Ruby file and its output:

```
PS D:\7th Sem\F1 - Ruby\Lab> ruby 21MIS1117_LAB1.rb
Common elements between the two arrays are: [4, 5]
PS D:\7th Sem\F1 - Ruby\Lab>
```