

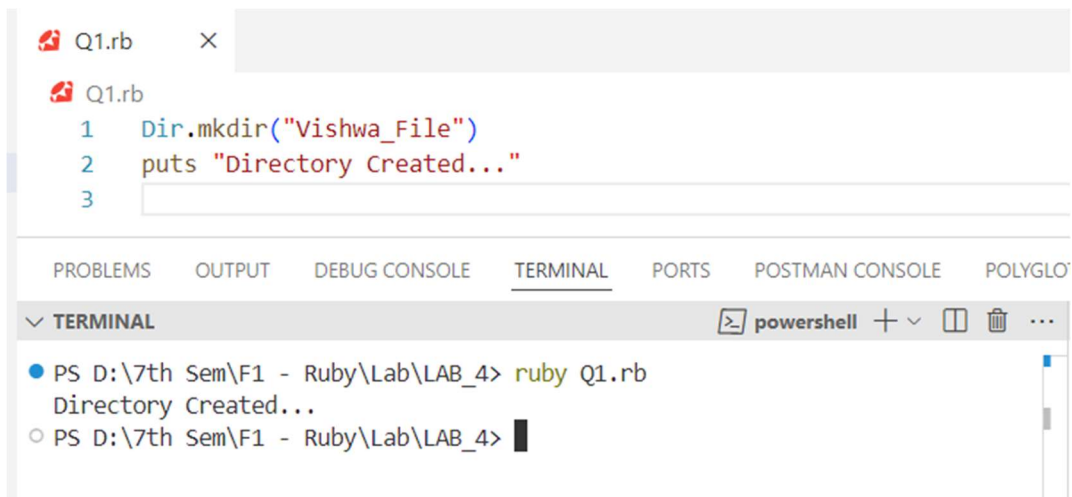
RUBY LAB -4

Name: Vishwanth P

Register No: 21MIS1117

Assessment 4

1. Create a directory in Ruby

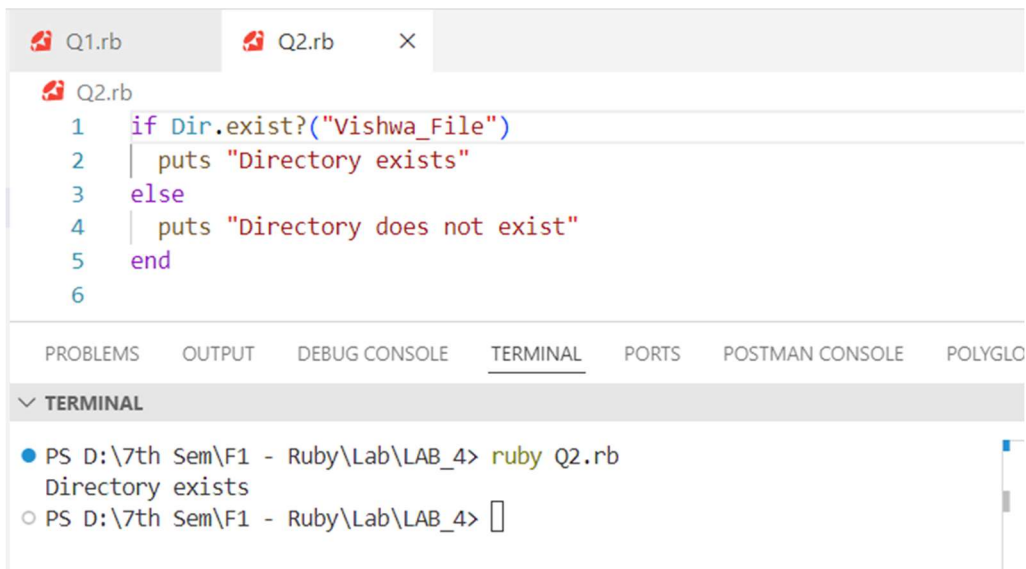


The screenshot shows an IDE with a file named Q1.rb open. The code in the file is:

```
1 Dir.mkdir("Vishwa_File")
2 puts "Directory Created..."
3
```

Below the code editor, the terminal window is active, showing the command `ruby Q1.rb` being executed. The output of the command is "Directory Created...".

2. Check if a directory exists in Ruby

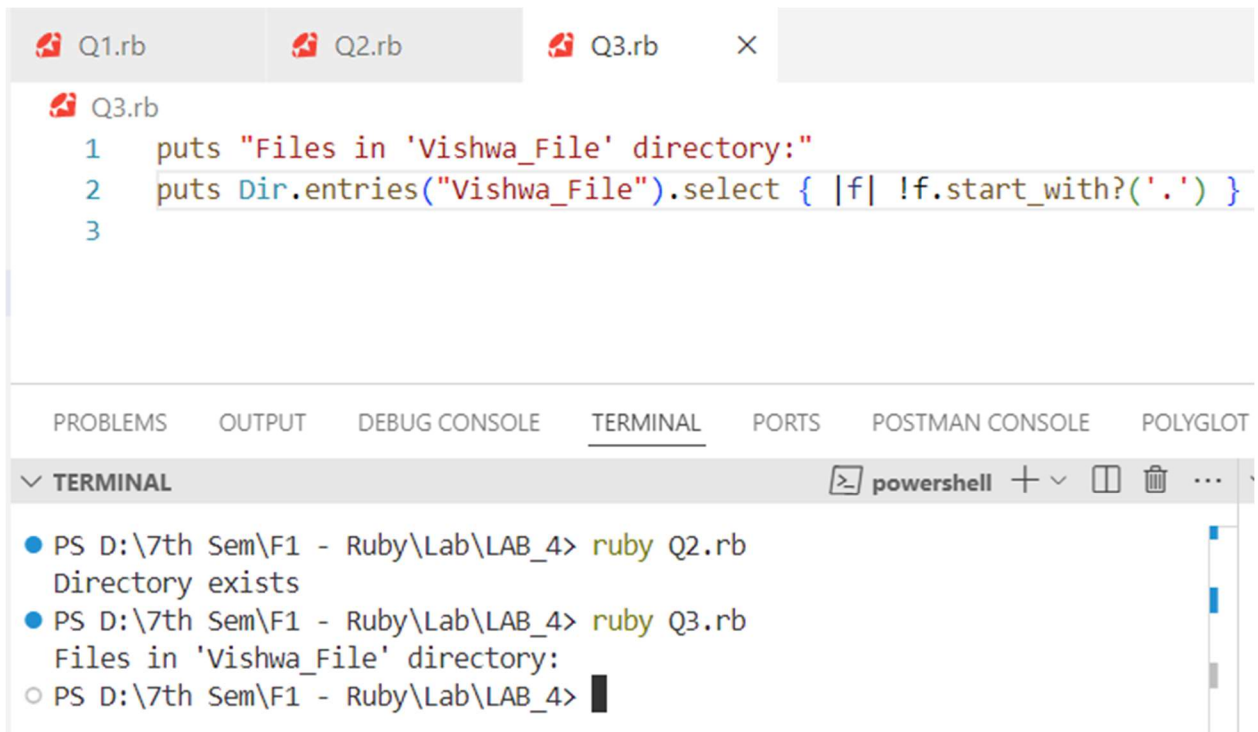


The screenshot shows an IDE with a file named Q2.rb open. The code in the file is:

```
1 if Dir.exist?("Vishwa_File")
2 | puts "Directory exists"
3 else
4 | puts "Directory does not exist"
5 end
6
```

Below the code editor, the terminal window is active, showing the command `ruby Q2.rb` being executed. The output of the command is "Directory exists".

3. List files in a directory using Ruby



The screenshot shows an IDE with three tabs: Q1.rb, Q2.rb, and Q3.rb. The Q3.rb tab is active, displaying the following Ruby code:

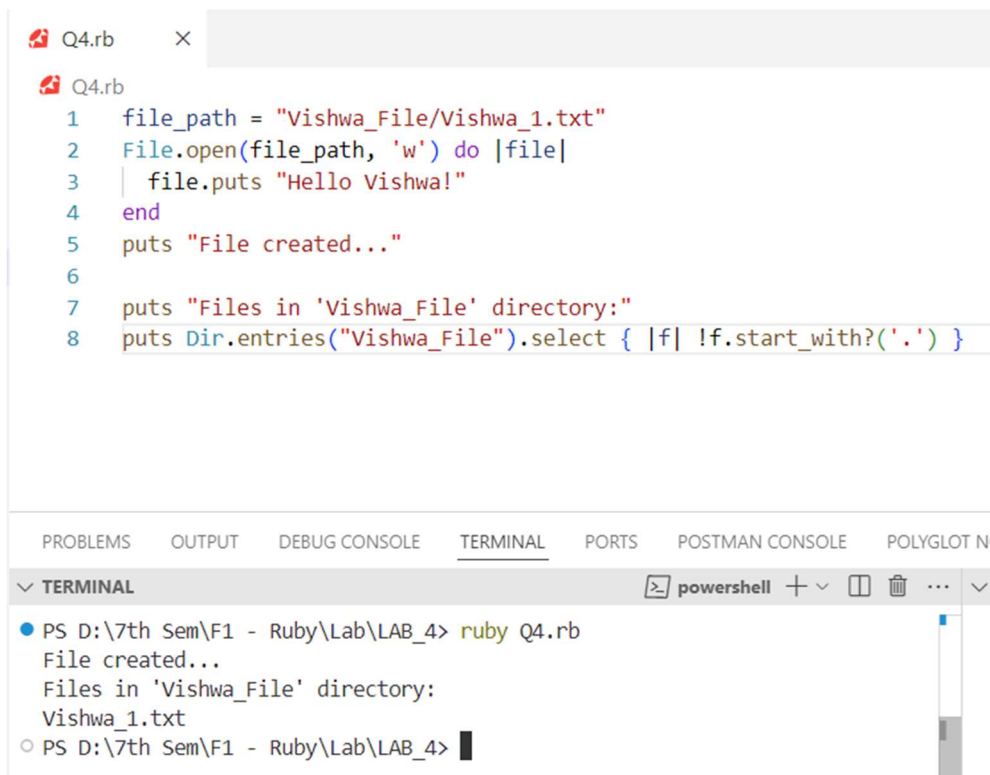
```
1 puts "Files in 'Vishwa_File' directory:"
2 puts Dir.entries("Vishwa_File").select { |f| !f.start_with?('.') }
3
```

Below the code editor is a terminal window titled "TERMINAL" with a PowerShell prompt. It shows the execution of two Ruby scripts:

- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q2.rb
Directory exists
- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q3.rb
Files in 'Vishwa_File' directory:

The terminal is currently waiting for the output of the second command.

4. Create a file in Ruby



The screenshot shows an IDE with a single tab: Q4.rb. The Q4.rb tab is active, displaying the following Ruby code:

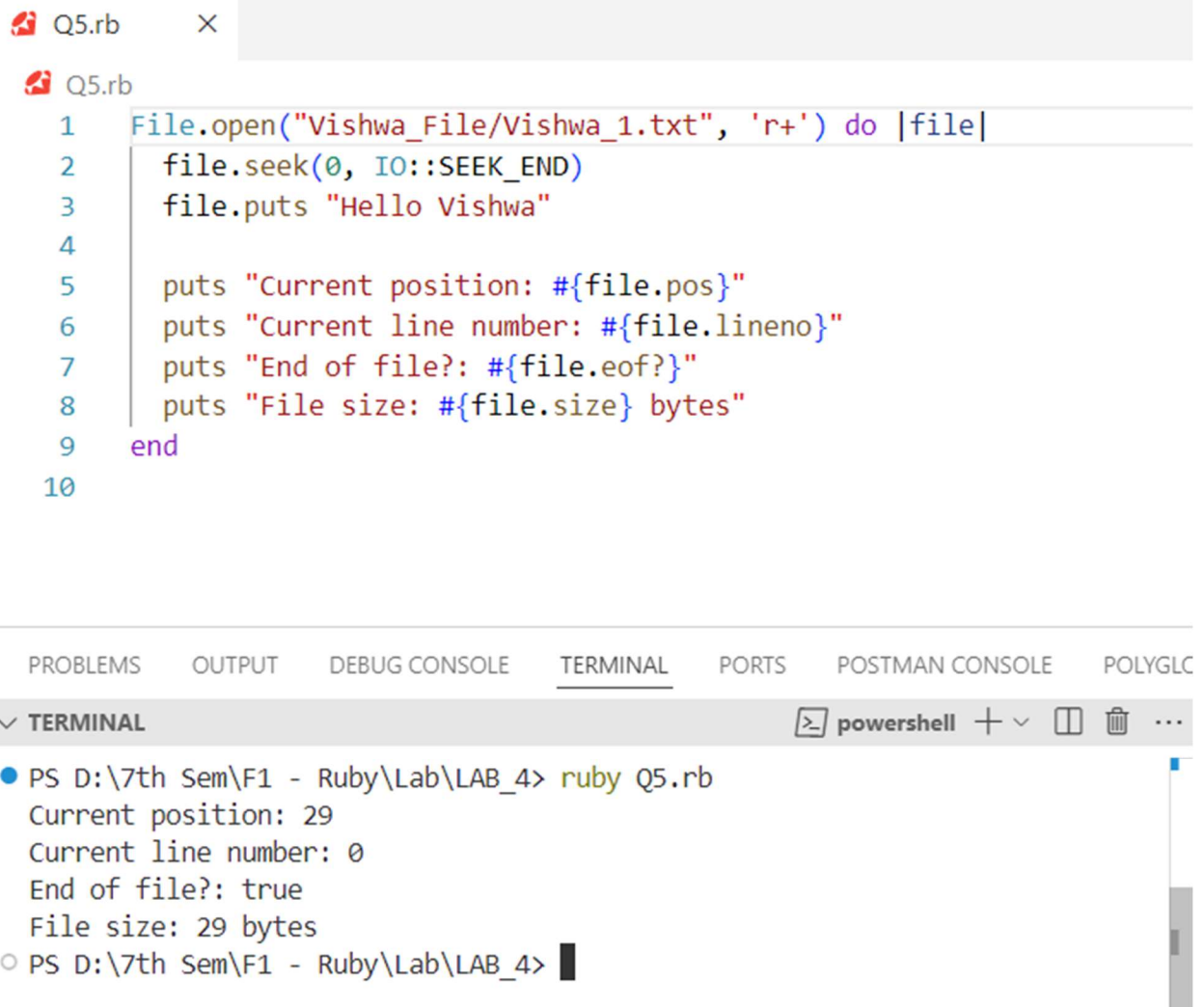
```
1 file_path = "Vishwa_File/Vishwa_1.txt"
2 File.open(file_path, 'w') do |file|
3   file.puts "Hello Vishwa!"
4 end
5 puts "File created..."
6
7 puts "Files in 'Vishwa_File' directory:"
8 puts Dir.entries("Vishwa_File").select { |f| !f.start_with?('.') }
```

Below the code editor is a terminal window titled "TERMINAL" with a PowerShell prompt. It shows the execution of the Ruby script Q4.rb:

- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q4.rb
File created...
Files in 'Vishwa_File' directory:
Vishwa_1.txt

The terminal is currently waiting for the output of the second command.

5. Write to a file in Ruby and use various methods like seek (), lineno(), eof(), size()



The screenshot shows an IDE with a Ruby file named `Q5.rb` and a terminal window. The Ruby code opens a file `Vishwa_File/Vishwa_1.txt` in append mode, seeks to the end, writes "Hello Vishwa", and then prints the current position, line number, end of file status, and file size. The terminal output shows the execution of `ruby Q5.rb` resulting in the expected values: position 29, line number 0, end of file true, and file size 29 bytes.

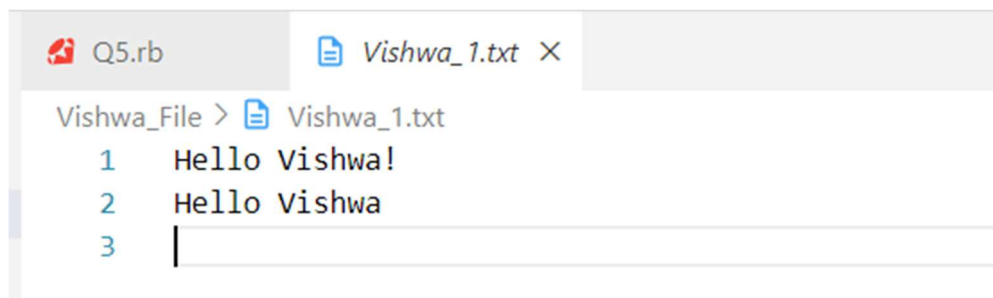
```
Q5.rb
1 File.open("Vishwa_File/Vishwa_1.txt", 'r+') do |file|
2   file.seek(0, IO::SEEK_END)
3   file.puts "Hello Vishwa"
4
5   puts "Current position: #{file.pos}"
6   puts "Current line number: #{file.lineno}"
7   puts "End of file?: #{file.eof?}"
8   puts "File size: #{file.size} bytes"
9 end
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGLC

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

- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q5.rb
Current position: 29
Current line number: 0
End of file?: true
File size: 29 bytes
- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> █

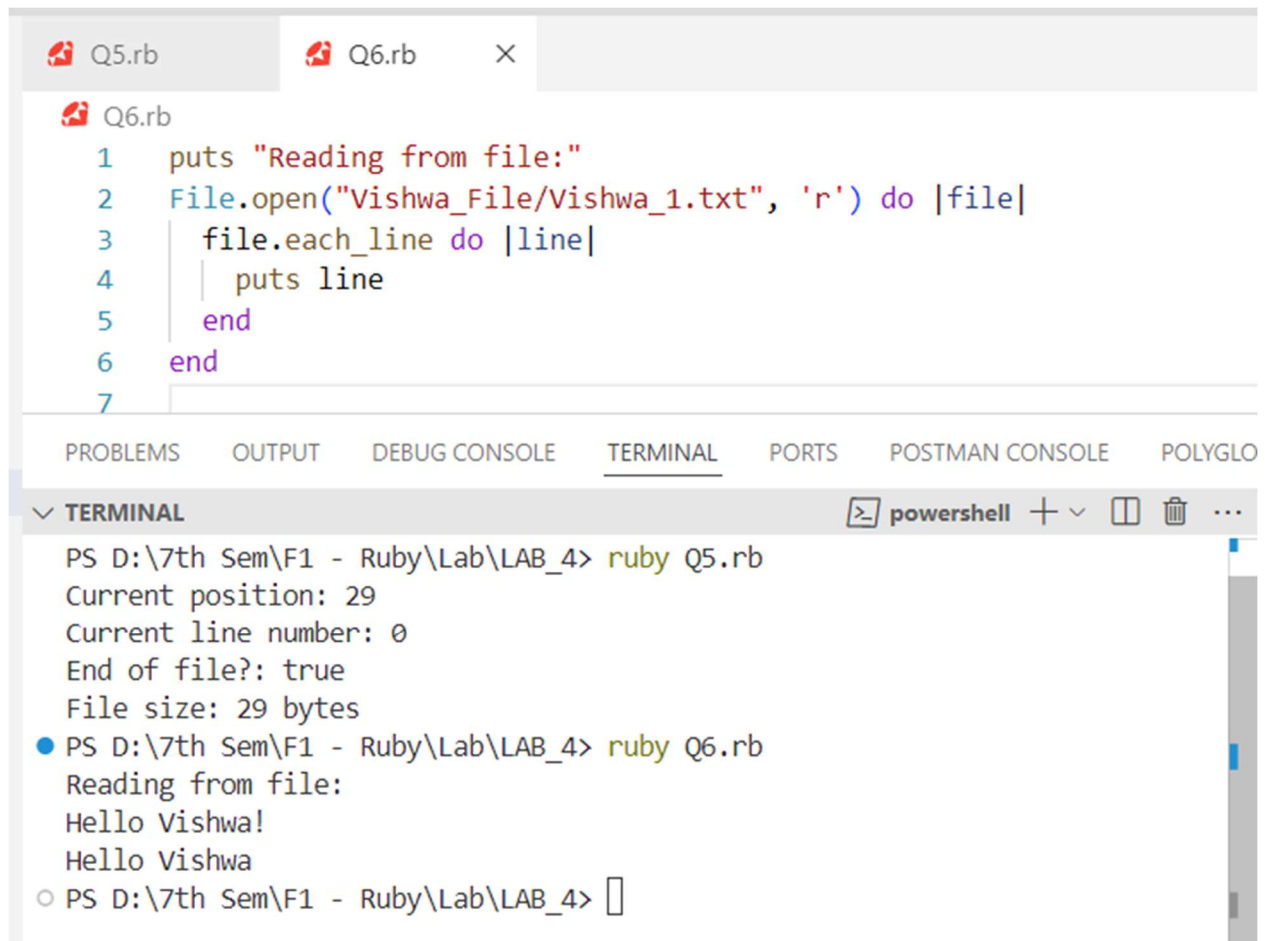
Vishwa_1.txt



The screenshot shows a text editor with two tabs: `Q5.rb` and `Vishwa_1.txt`. The `Vishwa_1.txt` tab is active, showing the file's contents: "Hello Vishwa!" on the first line and "Hello Vishwa" on the second line.

```
Vishwa_File > Vishwa_1.txt
1 Hello Vishwa!
2 Hello Vishwa
3 |
```

6. Read from a file in Ruby



The screenshot shows an IDE with two tabs: Q5.rb and Q6.rb. The Q6.rb tab is active, displaying the following Ruby code:

```
1 puts "Reading from file:"
2 File.open("Vishwa_File/Vishwa_1.txt", 'r') do |file|
3   file.each_line do |line|
4     puts line
5   end
6 end
7
```

Below the code editor is a terminal window titled "TERMINAL" with a PowerShell prompt. It shows the execution of two Ruby scripts:

```
PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q5.rb
Current position: 29
Current line number: 0
End of file?: true
File size: 29 bytes
● PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q6.rb
Reading from file:
Hello Vishwa!
Hello Vishwa
○ PS D:\7th Sem\F1 - Ruby\Lab\LAB_4>
```

7. Delete a file in Ruby



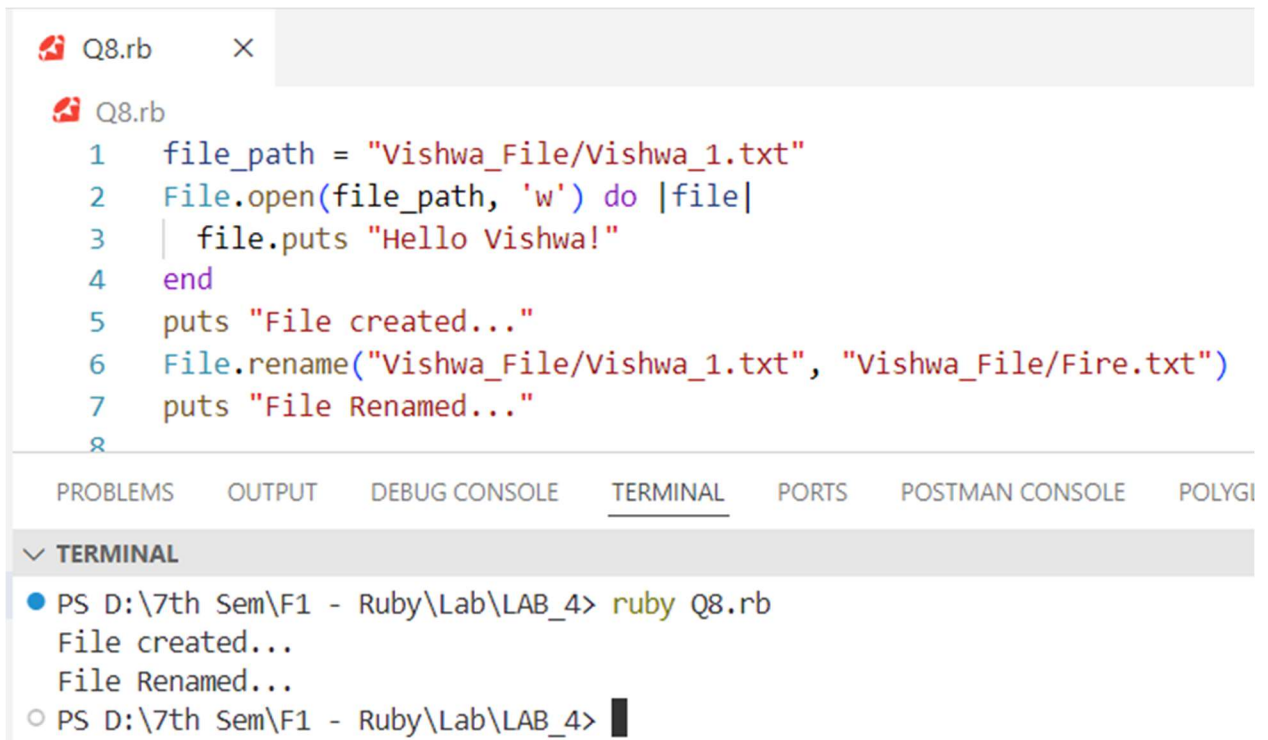
The screenshot shows an IDE with a single tab: Q7.rb. The Q7.rb tab is active, displaying the following Ruby code:

```
1 File.delete("Vishwa_File/Vishwa_1.txt") if File.exist?("Vishwa_File/Vishwa_1.txt")
2 puts "File deleted..."
3
```

Below the code editor is a terminal window titled "TERMINAL" with a PowerShell prompt. It shows the execution of the Ruby script:

```
● PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q7.rb
File deleted...
○ PS D:\7th Sem\F1 - Ruby\Lab\LAB_4>
```

8. Rename a file in Ruby



The screenshot shows an IDE with a file named Q8.rb. The code creates a file 'Vishwa_1.txt' with the text 'Hello Vishwa!' and then renames it to 'Fire.txt'. The terminal shows the execution of the script, outputting 'File created...' and 'File Renamed...'.

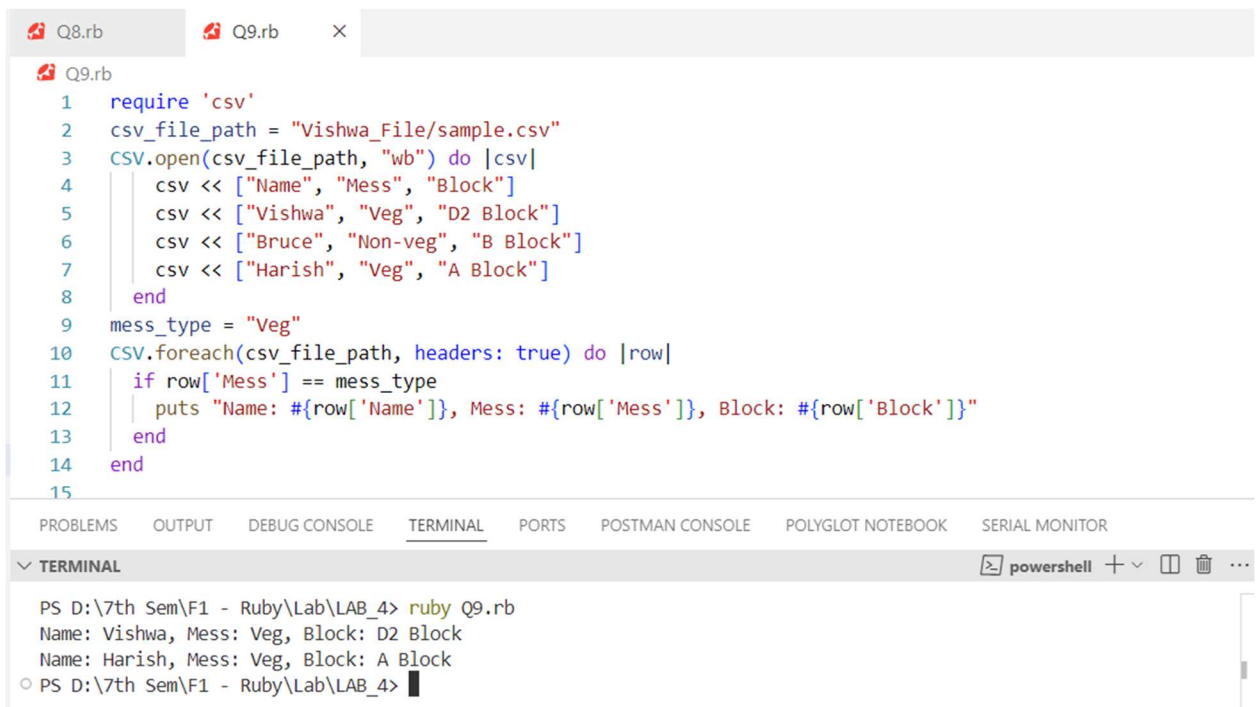
```
Q8.rb
1 file_path = "Vishwa_File/Vishwa_1.txt"
2 File.open(file_path, 'w') do |file|
3   file.puts "Hello Vishwa!"
4 end
5 puts "File created..."
6 File.rename("Vishwa_File/Vishwa_1.txt", "Vishwa_File/Fire.txt")
7 puts "File Renamed..."
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE POLYGI

✓ **TERMINAL**

- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q8.rb
File created...
File Renamed...
- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> █

9. Read a CSV and print specific rows and columns



The screenshot shows an IDE with a file named Q9.rb. The code reads a CSV file 'sample.csv' and prints the rows where the 'Mess' column is 'Veg'. The terminal shows the execution of the script, outputting the names and blocks of the rows where 'Mess' is 'Veg'.

```
Q9.rb
1 require 'csv'
2 csv_file_path = "Vishwa_File/sample.csv"
3 CSV.open(csv_file_path, "wb") do |csv|
4   csv << ["Name", "Mess", "Block"]
5   csv << ["Vishwa", "Veg", "D2 Block"]
6   csv << ["Bruce", "Non-veg", "B Block"]
7   csv << ["Harish", "Veg", "A Block"]
8 end
9 mess_type = "Veg"
10 CSV.foreach(csv_file_path, headers: true) do |row|
11   if row['Mess'] == mess_type
12     puts "Name: #{row['Name']}, Mess: #{row['Mess']}, Block: #{row['Block']}"
13   end
14 end
15
```

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

✓ **TERMINAL** powershell + ▢ ▢ ...

PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q9.rb
Name: Vishwa, Mess: Veg, Block: D2 Block
Name: Harish, Mess: Veg, Block: A Block
○ PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> █

10. File Splitting and Joining: Imagine you have a large file, such as a video or a database backup, that you need to transfer or store on multiple devices. However, transferring or storing the entire file at once may not be feasible due to limitations in file size or storage capacity. In this scenario, you can use a program that splits the large file into smaller parts and joins them back together when needed. How can you implement such a program using Ruby?

 Q10.rb

Codeium: Refactor | Explain | Generate Function Comment | X

```
1 def split_file(file_path, lines_per_file)
2   file_number = 1
3
4   File.open(file_path, "r") do |file|
5     while !file.eof?
6       File.open("split_file_part_#{file_number}.txt", "w") do |f|
7         lines_per_file.times do
8           line = file.gets
9           break if line.nil?
10          f.puts(line)
11        end
12      end
13      file_number += 1
14    end
15  end
16  puts "File split successfully into #{file_number - 1} parts."
17 end
18
```

Codeium: Refactor | Explain | Generate Function Comment | X

```
19 def join_files(output_file, total_parts)
20   File.open(output_file, "w") do |output|
21     (1..total_parts).each do |i|
22       File.open("split_file_part_#{i}.txt", "r") do |f|
23         output.puts(f.read)
24       end
25     end
26   end
27   puts "Files joined successfully into #{output_file}."
28 end
29
30 split_file("Vishwa_File/Vishwa.txt", 6)
31 join_files("Vishwa_File/Output.txt", 2)
--
```


Vishwa.txt

Q8.rb

Q9.rb

Q10.rb

Vishwa.txt X

Vishwa_File > Vishwa.txt

1 Biryani: Where Every Grain Tells a Story!

2 Life's Too Short for Bad Biryani!

3 Biryani - Happiness Served on a Plate!

4 Got Problems? Biryani's the Answer!

5 Spice Up Your Life, One Biryani at a Time!

6 Biryani: The Ultimate Mood Lifter!

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

POS

▼ TERMINAL

PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q10.rb
File split successfully into 1 parts.
Files joined successfully into Vishwa_File/Output.txt.
PS D:\7th Sem\F1 - Ruby\Lab\LAB_4>

Output.txt

Q8.rb

Q9.rb

Q10.rb

Output.txt X

Vishwa_File > Output.txt

1 Biryani: Where Every Grain Tells a Story!

2 Life's Too Short for Bad Biryani!

3 Biryani - Happiness Served on a Plate!

4 Got Problems? Biryani's the Answer!


5 Spice Up Your Life, One Biryani at a Time!

6 Biryani: The Ultimate Mood Lifter!

7 Biryani: The Ultimate Mood Lifter!

8 |

11. File Backup and Restore: Imagine you have important files or directories on your computer that you want to protect against data loss due to hardware failure, malware, or accidental deletion. In this scenario, you can use a program that creates a backup of the files or directories and restores them later if needed. How can you implement such a program using Ruby?

 Q11.rb

```
1  require 'fileutils'
2  require 'date'
3
4  Codeium: Refactor | Explain | Generate Function Comment | X
5  def backup(src, backup_dir)
6    ts = DateTime.now.strftime('%Y%m%d_%H%M%S')
7    bk_path = File.join(backup_dir, "#{File.basename(src)}_#{ts}")
8
9    FileUtils.cp_r(src, bk_path)
10   puts "Backup created at: #{bk_path}"
11 end
12
13 Codeium: Refactor | Explain | Generate Function Comment | X
14 def restore(bk_path, restore_dir)
15   orig_path = File.join(restore_dir, File.basename(bk_path).sub(/_\.d+/, ''))
16
17   if Dir.exist?(bk_path) || File.exist?(bk_path)
18     FileUtils.cp_r(bk_path, orig_path)
19     puts "Restored from: #{bk_path} to #{orig_path}"
20   else
21     puts "Backup not found!"
22   end
23 end
24
25 backup_dir = 'bups'
26 FileUtils.mkdir_p(backup_dir)
27
28 src = 'Vishwa_File/Vishwa.txt'
29 backup(src, backup_dir)
30
31 bk_path = Dir.glob("#{backup_dir}/#{File.basename(src)}_*").last
32 restore(bk_path, File.dirname(src)) if bk_path
```


Backup File (Vishwa.txt_20240930_200043)


 Vishwa.txt_20240930_200043 ×

bups >  Vishwa.txt_20240930_200043

- 1 Biryani: Where Every Grain Tells a Story!
- 2 Life's Too Short for Bad Biryani!
- 3 Biryani - Happiness Served on a Plate!
- 4 Got Problems? Biryani's the Answer!
- 5 Spice Up Your Life, One Biryani at a Time!
- 6 Biryani: The Ultimate Mood Lifter!

Restored File (Vishwa.txt_200043)

 Vishwa.txt_20240930_200043

 Vishwa.txt_200043 ×

Vishwa_File >  Vishwa.txt_200043

- 1 Biryani: Where Every Grain Tells a Story!
- 2 Life's Too Short for Bad Biryani!
- 3 Biryani - Happiness Served on a Plate!
- 4 Got Problems? Biryani's the Answer!
- 5 Spice Up Your Life, One Biryani at a Time!
- 6 Biryani: The Ultimate Mood Lifter!

12. File Renaming and Copying: Imagine you have a file that you want to rename or copy to a different location, optionally with a new name. In this scenario, you can use a program that allows you to perform these operations easily and efficiently. How can you implement such a program using Ruby?

 Q12.rb


```
1  require 'fileutils'
2
3  Codeium: Refactor | Explain | Generate Function Comment | X
4  def rename_file(old_name, new_name)
5      if File.exist?(old_name)
6          FileUtils.mv(old_name, new_name)
7          puts "Renamed '#{old_name}' to '#{new_name}'"
8      else
9          puts "File '#{old_name}' not found!"
10     end
11 end
12
13 Codeium: Refactor | Explain | Generate Function Comment | X
14 def copy_file(src, dest, new_name = nil)
15     if File.exist?(src)
16         copy_name = new_name || File.basename(src)
17         FileUtils.cp(src, File.join(dest, copy_name))
18         puts "Copied '#{src}' to '#{File.join(dest, copy_name)}'"
19     else
20         puts "File '#{src}' not found!"
21     end
22 end
23
24 old_file_name = 'Vishwa_File/Vishwa.txt'
25 new_file_name = 'Vishwa_File/Vishwa_new.txt'
26 destination = 'copies'
27
28 FileUtils.mkdir_p(destination)
29
30 rename_file(old_file_name, new_file_name)
31 copy_file(new_file_name, destination)
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    POSTMAN CONS
```

```
✓ TERMINAL
```

```
PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q12.rb
Renamed 'Vishwa_File/Vishwa.txt' to 'Vishwa_File/Vishwa_new.txt'
Copied 'Vishwa_File/Vishwa_new.txt' to 'copies/Vishwa_new.txt'
PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> █
```

13. File Searching and Filtering: Imagine you have a large collection of files on your computer and you want to find specific files based on their name, extension, size, or other criteria. In this scenario, you can use a program that searches for files matching a pattern or filters files based on certain criteria. How can you implement such a program using Ruby?

 Q13.rb

```
Codeium: Refactor | Explain | Generate Function Comment | X
1  def find_files(dir, name_pat: nil, ext: nil, min_size: nil)
2    files = Dir.glob("#{dir}/**/*").select { |f| File.file?(f) }
3    files.select! { |f| File.basename(f).include?(name_pat) } if name_pat
4    files.select! { |f| File.extname(f) == ext } if ext
5    files.select! { |f| File.size(f) >= min_size } if min_size
6    files
7  end
8
9  dir = '.'
10 name_pat = 'Vishwa'
11 ext = '.txt'
12 min_size = 100
13
14 results = find_files(dir, name_pat: name_pat, ext: ext, min_size: min_size)
15
16 if results.empty?
17   puts "No files found."
18 else
19   puts "Found files:"
20   results.each { |f| puts f }
21 end
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

▼ TERMINAL

PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q13.rb
○ Found files:
  ./Vishwa_File/Vishwa_new.txt
  ./copies/Vishwa_new.txt
PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> 
```


14. File Sorting and Merging: Imagine you have multiple files that contain data that needs to be combined into a single file, such as log files from different servers or reports from multiple departments. In this scenario, you can use a program that sorts the files based on certain criteria and merges them into a single file. How can you implement such a program using Ruby?

 Q14.rb

```
Codeium: Refactor | Explain | Generate Function Comment | X
1  def merge_and_sort_files(file_paths, output_file)
2      all_data = []
3
4      file_paths.each do |file|
5          if File.exist?(file)
6              all_data.concat(File.readlines(file).map(&:strip))
7          else
8              puts "File '#{file}' not found!"
9          end
10     end
11
12     sorted_data = all_data.uniq.sort
13
14     File.open(output_file, 'w') do |f|
15         sorted_data.each { |line| f.puts(line) }
16     end
17
18     puts "Merged and sorted data written to '#{output_file}'"
19 end
20
21 file_paths = ['file1.txt', 'file2.txt', 'file3.txt']
22 output_file = 'merged_output.txt'
23
24 merge_and_sort_files(file_paths, output_file)
25
```

Output

- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> ruby Q14.rb
Merged and sorted data written to 'merged_output.txt'
- PS D:\7th Sem\F1 - Ruby\Lab\LAB_4> █

 file1.txt ×

 file1.txt

```
1 Junior Kuppanna - Trichy
2 Dindugal Thalakatti - Dindugal
~
```

 file2.txt ×


 file2.txt


```
1 Anjappar - Madurai
2 SS Hyderabad - Chennai
```

 file3.txt ×

 file3.txt

```
1 Pandiyan Hotel - Madurai
2 Khalid's Biryani - Trichy
```

 merged_output.txt ×

 merged_output.txt

```
1
2 Anjappar - Madurai
3 Dindugal Thalakatti - Dindugal
4 Junior Kuppanna - Trichy
5 Khalid's Biryani - Trichy
6 Pandiyan Hotel - Madurai
7 SS Hyderabad - Chennai
8
```