Ruby Lab Assessment 4:

21MIS1021
VIMAL KUMAR S

1. Create a directory in Ruby

lab

File Edit View

```
dir_name = "21MIS1021"
Dir.mkdir(dir_name)
```

Start Command Prompt with Ruby

```
C:\Users\student.PROGRAMMING404>cd C:\Users\student.PROGRAMMING404\Desktop\21MIS1021

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

2. Check if a directory exists in Ruby

lab

File Edit View

```
dir_name = "21MIS1021"
if Dir.exist?(dir_name)
puts "The directory #{dir_name} exists."
else
puts "The directory #{dir_name} does not exist."
end
```

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb
The directory 21MIS1021 exists.

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

3. List files in a directory using Ruby

```
dir_name = "21MIS1021"
files = Dir.entries(dir_name)
files.each do |file|
puts file
end
```

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb
.
..
```

4. Create a file in Ruby

```
file_name = "vimal.rb"
File.new(file_name, "w")
```

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

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

```
file_name = "vimal.rb"
File.open(file_name, "w") do |file|
file.puts "Hello, world!"
file.puts "This is a new line. "
file.print "This is "
file.puts "another line."
end

file_name = "vimal.rb"
File.open(file_name, "r") do |file|
puts file.size
puts file.lineno
puts file.eof?
puts file.seek(0)
end
```

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb
60
0
false
0
```

6. Read from a file in Ruby

lab

File Edit View

```
file_name = "vimal.rb"
File.open(file_name, "r") do |file|
line = file.readlines
line.each do |line|
puts line
end
```

Ln 3, Col 22 | 100% | Windows (CRLF) | UTF-

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb
Hello, world!
This is a new line.
This is another line.

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

7. Delete a file in Ruby

lab

File Edit View

```
file_name = "vimal.rb"
File.delete(file_name)
```

Ln 2, Col 3 | 110% | Windows (CRLF) | UT

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

8. Rename a file in Ruby

```
lab
File  Edit  View

old_name = "vimal.rb"
new_name = "kumar.rb"
File.rename(old_name, new_name)

Ln 3, Col 32   110%   Windows (CRLF)   UTF-8

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

9. Read a CSV and print specific rows and columns

```
lab
File  Edit  View

require 'csv'
csv_file = "data1.csv"
CSV.foreach(csv_file) do |row|
puts row[0]
puts row[2]

if row[0] == "vimal"
puts row.join(",")
end
end

Start Command Prompt with Ruby

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb
vimal
vimal
vimal,vimal,vimal,vimal,vimal,vimal,vimal,vimal,vimal,vimal
sanjay
sanjay
poovi
poovi
maneeth
maneeth
anjali
anjali
kavya
kavya
nivedha
nivedha
pooja
pooja
tracita
tracita
kribha
kribha

C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>
```

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?

lab

File Edit View

```
def split_file(input_file, chunk_size)
base_name = File.basename(input_file)
output_dir = File.dirname(input_file)

File.open(input_file, "rb") do |input|
counter = 1
while chunk = input.read(chunk_size)
chunk_file = File.join(output_dir,"#{base_name}.part#{counter}")
File.open(chunk_file, "wb") {
|output| output.write(chunk) }
counter += 1
end
end

puts"File split complete!"
end

def join_files(output_file)
base_name = File.basename(output_file)
output_dir = File.dirname(output_file)

Dir.chdir(output_dir) do
chunks = Dir.glob("#{base_name}.part*").sort
File.open(output_file, "wb") do
|output|
chunks.each do |chunk_file|
File.open(chunk_file, "rb"){
|input| output.write(input.read)}
end
end
end

puts"File join complete!"
end

# Example usage:
#splitting a file into smaller parts
split_file("data1.csv", 1000000)

join_files("datanew.csv")
```

```
C:\Users\student.PROGRAMMING404\Desktop\21MIS1021>ruby lab.rb
File split complete!
File join complete!
```

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?

lab

File Edit View

```ruby
require 'fileutils'

def create_backup(source_path, backup_path)
  FileUtils.cp_r(source_path, backup_path)
  puts "Backup created successfully!"
rescue Errno::ENOENT
  puts "Source path not found!"
end

def restore_backup(backup_path, destination_path)
  FileUtils.cp_r(backup_path, destination_path)
  puts "Backup restored successfully!"
rescue Errno::ENOENT
  puts "Backup path not found!"
end

# Example usage
source_path = "data1.csv"
backup_path = "21MIS1021"
destination_path = "new file"

# Create a backup
create_backup(source_path, backup_path)

# Restore the backup
restore_backup(backup_path, destination_path)
```

```
C:\Users\Dell\Desktop\21MIS1021>ruby lab.rb
Backup created successfully!
Backup restored successfully!
```

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?

lab

File Edit View

```
require 'fileutils'

def rename_file(source_path, new_name)
  # Get the directory path and the current file name
  dir_path = File.dirname(source_path)
  base_name = File.basename(source_path)

  # Create the new file path by combining the directory path and the new name
  new_path = File.join(dir_path, new_name)

  # Rename the file
  FileUtils.mv(source_path, new_path)

  puts "File renamed successfully."
end

def copy_file(source_path, destination_path)
  # Copy the file to the destination path
  FileUtils.cp(source_path, destination_path)

  puts "File copied successfully."
end

# Example usage

# Rename a file
rename_file('dataset.txt', 'newdata.txt')

# Copy a file to a different location
copy_file('newdata.txt', 'Lab')
```

OUTPUT:

```
C:\Users\Dell\Desktop\21MIS1021>ruby lab.rb
File renamed successfully.
File copied successfully.
```

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?

```
def search_files(directory, pattern)
  matching_files = []
   Dir.glob("#{directory}/**/#{pattern}") do |file_path|
    matching_files << file_path if File.file?(file_path)
  end
    matching_files
end
def filter_files(directory, criteria)
  filtered_files = []
    Dir.glob("#{directory}/**/*") do |file_path|
    if File.file?(file_path) && file_matches_criteria?(file_path, criteria)
      filtered_files << file_path
    end
  end
    filtered_files
end
def file_matches_criteria?(file_path, criteria)
  criteria.each do |key, value|
    case key
    when :name
      return false unless File.basename(file_path).include?(value)
    when :extension
      return false unless File.extname(file_path) == value
    when :size
      return false unless File.size(file_path) == value
    # Add more criteria checks as needed
    end
  end
    true
end
```

```
# Example usage
directory = "21MIS1021" # Replace with the actual directory path
pattern = "*.txt" # Replace with the desired file pattern
matching_files = search_files(directory, pattern)
puts "Matching files:"
matching_files.each { |file_path| puts file_path }
# Example criteria for filtering
criteria = {
  name: "VIMAL KUMAR S",
  extension: "newdata.txt",
  size: 1024
}
filtered_files = filter_files(directory, criteria)
puts "Filtered files:"
filtered_files.each { |file_path| puts file_path }
```

```
C:\Users\Dell\Desktop\21MIS1021>ruby lab.rb
Matching files:
Filtered files:
```

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?

lab

File Edit View

```ruby
def merge_files(file_paths, output_file)
  lines = []

  file_paths.each do |file_path|
    File.foreach(file_path) do |line|
      lines << line.chomp
    end
  end

  lines.sort!  # Sorting the lines based on certain criteria

  File.open(output_file, "w") do |file|
    lines.each do |line|
      file.puts line
    end
  end

  puts "File merged successfully"
end

# Example usage:
files_to_merge = ["newdata.txt", "newdata2.txt"]
output_file = "merged_file.txt"
merge_files(files_to_merge, output_file)
```

```
C:\Users\Dell\Desktop\21MIS1021>ruby lab.rb
File merged successfully
```