**EASTERN INTERNATIONAL UNIVERSITY** **Practice Assignment – Quarter 2, 2024-2025**

**SCHOOL OF COMPUTING Course Name:** Special Topic 1

**AND INFORMATION TECHNOLOGY** **Course Code:** CSE 484

🙙🕮🙛 **Student’s Full Name:** Ha Quang Minh

**Student ID: 1931220012**

**Lab 2**

**Background**

**Required Resources:**

* 1 PC with operating system of your choice
* VSCode installed with appropriate python interpreter
* CSR1000V can ping your host computer

**Questions:**

**Part 1: General Function and File**

# Paint job estimator

A painting company has determined that for every 112 square feet of wall space, one gallon of paint and eight hours of labor will be required. The company charges $35.00 per hour for labor. Write a program that asks the user to enter the square feet of wall space to be painted and the price of the paint per gallon. The program should display the following data:

* The number of gallons of paint required
* The hours of labor required
* The cost of the paint
* The labor charges
* The total cost of the paint job

# Monthly sales tax

A retail company must file a monthly sales tax report listing the total sales for the month, and the amount of state and county sales tax collected. The state sales tax rate is 5 percent and the county sales tax rate is 2.5 percent. Write a program that asks the user to enter the total sales for the month. From this figure, the application should calculate and display the following:

* The amount of county sales tax
* The amount of state sales tax
* The total sales tax (county plus state)

# Rock, paper, scissors game

Write a program that lets the user play the game Rock, Paper, Scissors against the computer. The program should work as follows:

1. When the program begins, a random number in the range of 1 through 3 is generated. If the number is 1, then the computer has chosen rock. If the number is 2, then the computer has chosen paper. If the number is 3, then the computer has chosen scissors. (Don’t display the computer’s choice yet.)
2. The user enters his or her choice of “rock,” “paper,” or “scissors” at the keyboard.
3. The computer’s choice is displayed.
4. A winner is selected according to the following rules:

* If one player chooses rock and the other player chooses scissors, then rock wins.
* (Rock smashes scissors.)
* If one player chooses scissors and the other player chooses paper, then scissors wins.
* (Scissors cuts paper.)
* If one player chooses paper and the other player chooses rock, then paper wins.
* (Paper wraps rock.)
* If both players make the same choice, the game must be played again to determine
* the winner.

# File head display

Write a program that asks the user for the name of a file. The program should display only the first five lines of the file’s contents. If the file contains less than five lines, it should display the file’s entire contents.

# High score

Assume that a file named scores.txt exists on the computer’s disk. It contains a series of records, each with two fields – a name, followed by a score (an integer between 1 and 100). Write a program that displays the name and score of the record with the highest score, as well as the number of records in the file. (Hint: Use a variable and an “if” statement to keep track of the highest score found as you read through the records, and a variable to keep count of the number of records.)

# Word list file writer

Write a program that asks the user how many words they would like to write to a file, and then asks the user to enter that many words, one at a time. The words should be written to a file.

# Word list file reader

This exercise assumes you have completed Programming Exercise 6, Word List File Writer. Write another program that reads the words from the file and displays the following data:

* The number of words in the file.
* The longest word in the file.
* The average length of all the words in the file.

# Personal web page generator

Write a program that asks the user for his or her name, then asks the user to enter a sentence that describes himself or herself. Here is an example of the program’s screen:

**Enter your name**: Julie Taylor **Enter**

**Describe yourself**: I am a computer science major, a member of the Jazz club, and I hope to work as a mobile app developer after I graduate. **Enter**

Once the user has entered the requested input, the program should create an HTML file, containing the input, for a simple Web page. Here is an example of the HTML content, using the sample input previously shown:

<html>

<head>

</head>

<body>

<center>

<h1>Julie Taylor</h1>

</center>

<hr />

I am a computer science major, a member of the Jazz club,

and I hope to work as a mobile app developer after I graduate.

<hr />

</body>

</html>

**Part 2: File Reading and Configuration File Generation**

# Reading csv data and generating configuration files for network devices

## Theoretical overview

* CSV (Comma-Separated Values) is a simple, tabular format commonly used for structured data. It allows easy sharing of data across applications. Its structure consists of rows and columns separated by commas with first row often containing headers (field names).

Rows

* An example of CSV:

hostname,ip\_address,domain\_name,username,password Router1,192.168.1.1,mydomain.com,admin,cisco123 Switch1,192.168.1.2,mydomain.com,admin,cisco123

* Python CSV Module:
  + The csv module provides functionality to read and write CSV files.
  + csv.DictReader: Reads CSV data into dictionaries where the keys are the headers.

## Tasks:

1. Open a folder in VSCode
2. Copy the example content and paste to a file named devices.csv in the folder
3. Run the following code:

import csv

def read\_device\_data(csv\_file):

"""

Read device information from a CSV file.

:param csv\_file: Path to the CSV file

:return: List of dictionaries with device data

"""

devices = []

with open(csv\_file, "r") as file:

reader = csv.DictReader(file)

for row in reader:

devices.append(row)

return devices

def generate\_config(device):

"""

Generate configuration commands for a device.

:param device: Dictionary with device information

:return: List of configuration commands

"""

config = [

f"hostname {device['hostname']}",

f"ip domain-name {device['domain\_name']}",

"crypto key generate rsa",

"1024",

f"username {device['username']} privilege 15 secret {device['password']}",

"line vty 0 4",

"transport input ssh",

"login local",

"exit",

"ip ssh version 2",

]

return config

def write\_config\_to\_file(config, filename):

"""

Write configuration commands to a file.

:param config: List of configuration commands

:param filename: Path to the output file

"""

with open(filename, "w") as file:

file.write("\n".join(config))

print(f"Configuration written to {filename}")

# Read devices from CSV and generate configurations

devices = read\_device\_data("devices.csv")

for device in devices:

config = generate\_config(device)

output\_file = f"{device['hostname']}\_config.txt"

write\_config\_to\_file(config, output\_file)

Capture the result to your report.

1. Install netmiko and paramiko modules (using pip install netmiko paramiko) to your VSCode
2. Verify the ssh connectivity to your CSR1000V router
3. Run the following code, **modify some information as needed**:

from netmiko import ConnectHandler

def deploy\_config\_to\_device(router, config\_file):

"""

Deploy a configuration file to a network device.

:param router: Dictionary containing router connection details

:param config\_file: Path to the configuration file

:return: Output from the device

"""

try:

# Read configuration from file

with open(config\_file, "r") as file:

config\_lines = file.readlines()

# Establish SSH connection

connection = ConnectHandler(\*\*router)

connection.enable()

# Send configuration to the device

print(f"Deploying configuration to {router['host']}...")

output = connection.send\_config\_set(config\_lines)

# Save configuration

connection.save\_config()

connection.disconnect()

return output

except Exception as e:

return f"Error deploying configuration: {e}"

# Replace with your router's details

router\_details = {

"device\_type": "cisco\_ios",

"host": "192.168.1.1",

"username": "admin",

"password": "cisco123",

"secret": "enable123", # Omit if no enable password is needed

}

# Path to the configuration file

config\_file\_path = "Router1\_config.txt"

# Deploy the configuration

output = deploy\_config\_to\_device(router\_details, config\_file\_path)

print(output)

Capture the result to your report.

# JSON to configuration file

## ****Theoretical overview:****

1. **Purpose of JSON**:
   * JSON (JavaScript Object Notation) is a lightweight format for representing structured data.
   * Commonly used in APIs and configuration files.
2. **JSON vs CSV**:
   * JSON supports nested structures (hierarchical data), while CSV is flat.
   * JSON is more flexible but slightly harder to parse.
3. **Python JSON Module**:
   * The json module is used to parse JSON files.
   * json.load(): Reads a JSON file and converts it into a Python data structure (list, dictionary, etc.).
4. **Network Configuration**:
   * The same principles of configuration generation apply, but JSON allows for more complex data representation.

## Tasks:

1. Copy the following content to a file named devices.json in your folder

[

{

"hostname": "Router1",

"ip\_address": "192.168.1.1",

"domain\_name": "mydomain.com",

"username": "admin",

"password": "cisco123"

},

{

"hostname": "Switch1",

"ip\_address": "192.168.1.2",

"domain\_name": "mydomain.com",

"username": "admin",

"password": "cisco123"

}

]

1. Copy and run the following code, then capture the result content to your report:

import json

def read\_device\_data(json\_file):

"""

Read device information from a JSON file.

:param json\_file: Path to the JSON file

:return: List of dictionaries with device data

"""

with open(json\_file, "r") as file:

return json.load(file)

def generate\_config(device):

"""

Generate configuration commands for a device.

:param device: Dictionary with device information

:return: List of configuration commands

"""

config = [

f"hostname {device['hostname']}",

f"ip domain-name {device['domain\_name']}",

"crypto key generate rsa",

"1024",

f"username {device['username']} privilege 15 secret {device['password']}",

"line vty 0 4",

"transport input ssh",

"login local",

"exit",

"ip ssh version 2",

]

return config

def write\_config\_to\_file(config, filename):

"""

Write configuration commands to a file.

:param config: List of configuration commands

:param filename: Path to the output file

"""

with open(filename, "w") as file:

file.write("\n".join(config))

print(f"Configuration written to {filename}")

# Read devices from JSON and generate configurations

devices = read\_device\_data("devices.json")

for device in devices:

config = generate\_config(device)

output\_file = f"{device['hostname']}\_config.txt"

write\_config\_to\_file(config, output\_file)

**Submission**: Must include:

* a **pdf** report file containing your information (student id, name), and images of the diagrams, codes, answers, evident,…
* and all source **code files** (if any)

in a **zipped** (.zip or \*.rar) file to Moodle