Lab 10 Guide:

First, create the troubleshoot.py file that will run all the troubleshooting code. Code will be explained in the video, but it is located in Lab4/troubleshoot.py

```
import os
import re
import pyshark
from csv import DictWriter
import subprocess
import time
import csv
from netmiko import ConnectHandler

def getSyslog():
    file = "traps.pcap"
    cap = pyshark.FileCapture(file)
    traps = []
    i = 1
    field_names = ['ID', 'Router', 'Message', 'Level']
    with open('Syslog.csv', 'w') as f_object:
        for packet in cap:
            dictwriter_object = DictWriter(f_object, field_names)
            if 'Syslog' in packet and packet.Syslog.level <= "5":
                traps.append(str(packet.Syslog))
                #match = re.search(r'Syslog message id:\s*(.*)',
str(packet.Syslog))
                match = re.search(r'%(.*)', str(packet.Syslog))
                if not match:
                    continue
                result = match.group(1)
                syslog_dict = {'ID':i, 'Router': packet.Syslog.hostname,
'Message':result, 'Level': packet.Syslog.level}
                dictwriter_object.writerow(syslog_dict)
            i+=1

#getSyslog()

def extract_interface_states(csv_file):
    extracted_data = []  # List to store extracted interface and state
    interface_pattern = re.compile(r'Interface Ethernet(\S+),', re.IGNORECASE)
    state_pattern = re.compile(r'state\s+to\s+(up|down)', re.IGNORECASE)

    with open(csv_file, 'r') as file:
        reader = csv.reader(file)
```

```
        for row in reader:
            if len(row) < 3:  # Skip rows without enough data
                continue

            # Extract the message field from the row
            message = row[2]

            # Search for the interface and state
            interface_match = interface_pattern.search(message)
            state_match = state_pattern.search(message)

            if interface_match:
                interface = interface_match.group(1)
                state = state_match.group(1) if state_match else "unknown"
                extracted_data.append({"Interface": interface, "State": state,
"Hostname": row[1]})

    return extracted_data


def sshInfo():
    csv_file =
"/home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab4/devices.csv"
    data = {}

    with open(csv_file, "r") as file:
        reader = csv.DictReader(file)
        for row in reader:
            router_name = row["hostname"]
            router_data = {
                "device_type": row["device_type"],
                "ip": row["ip"],
                "username": row["username"],
                "password": row["password"]
            }
            data[router_name] = router_data

    return data

def config_interface(ip, user, password, interface):

    arista_device = {
        'device_type': "arista_eos",
        'host': ip,  # Replace with the device IP address
        'username': user,    # Replace with your username
```

```
        'password': password, # Replace with your password
    }

    try:
        # Establish the connection
        connection = ConnectHandler(**arista_device)
        print(f"Connected to {arista_device['host']}")

        # Enter enable mode (if required)
        if connection.check_enable_mode() is False:
            connection.enable()

        # Send the command to configure the interface
        commands = [
            'interface ethernet'+interface,  # Enter interface configuration mode
            'no shutdown'           # Bring up the interface
        ]
        output = connection.send_config_set(commands)

        # Print the command output
        print("Configuration output:")
        print(output)

        # Close the connection
        connection.disconnect()
        print("Connection closed.")

    except Exception as e:
        print(f"An error occurred: {e}")

def interface_no_shut(interface, hostname):
    routers = sshInfo()
    extract_interface_states("Syslog.csv")
    for i in routers:
        if hostname == i:
            config_interface(routers[i]['ip'], routers[i]['username'],
routers[i]['password'], interface)

def find_dst_ip(hostname, interface):
    csv_file =
"/home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab4/devices.csv"

    routers = sshInfo()
    for i in routers:
        if hostname == i:
```

```
            ip = routers[i]['ip']
            username = routers[i]['username']
            password = routers[i]['password']

    device = {
        'device_type': "arista_eos",
        'host': ip,
        'username': username,
        'password': password
    }

    command = "show ip int br"

    with ConnectHandler(**device) as net_connect:
        net_connect.enable()
        output = net_connect.send_command(command)
        print(output)

    match = re.search(r"Ethernet"+interface+"\s+(\S+)", output)

    if match:
        ip_address_with_subnet = match.group(1)
        ip_address = ip_address_with_subnet.split('/')[0]  # Split and get only
the IP address
        print(f"The IP address for Ethernet1.10 is: {ip_address}")
        return ip_address
    else:
        print("Ethernet1.10 not found")


def get_ip_connectivity(hostname, interface):
    csv_file =
"/home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab4/devices.csv"
    dst_ip = find_dst_ip(hostname, interface)
    devices = ["R1", "R2", "R3", "R4"]
    routers = sshInfo()
    ping_success = {}
    for i in devices:
        if hostname != i:
            ip = routers[i]['ip']
            username = routers[i]['username']
            password = routers[i]['password']

            device = {
                'device_type': "arista_eos",
```

```
                'host': ip,
                'username': username,
                'password': password
            }

            command = "ping "+str(dst_ip)

            with ConnectHandler(**device) as net_connect:
                net_connect.enable()
                output = net_connect.send_command(command)

            match = re.search(r'(\d+) packets transmitted, (\d+) received',
output)

            if match:
                packets_transmitted = int(match.group(1))
                packets_received = int(match.group(2))
                if packets_transmitted == packets_received:
                    ping_success[i] = "True"
                else:
                    ping_success[i] = "False"
            else:
                ping_success[i] = "False"
    return ping_success


def fix_interface_state():
    getSyslog()
    seen = set()
    data = extract_interface_states("Syslog.csv")

    logs = []

    logs.append("Checking for down interfaces\n")
    for entry in data:
        if entry['State'] == "down":
            if entry['Interface'] in seen:
                continue
            seen.add(entry['Interface'])
            logs.append("Found Interface: "+entry['Interface']+" to be down\n")
            logs.append("Now doing a no shutdown on Interface:
"+entry['Interface']+"\n")
            interface_no_shut(entry['Interface'], entry['Hostname'])
    time.sleep(1)
    getSyslog()
```

```
    data_after = extract_interface_states("Syslog.csv")
    seen_after = set()
    logs.append("Now checking in Syslogs that interface is up\n")
    for entry in data_after:
        if entry['State'] == "up":
            if entry['Interface'] in seen_after:
                continue
            seen_after.add(entry['Interface'])
            logs.append("Interface: "+entry['Interface']+ " is now up\n")

    logs.append("Now checking IP connectivity from other devices\n")

    seen_ip = set()
    for entry in data_after:
        if entry['State'] == "up":
            if entry['Interface'] in seen_ip:
                continue
            seen_ip.add(entry['Interface'])
            pings = get_ip_connectivity(entry['Hostname'], entry['Interface'])

            for i in pings:
                if pings[i] == "True":
                    logs.append(i+" ping PASSED for device: "+entry['Hostname']+"
at interface: "+entry['Interface']+"\n")
                else:
                    logs.append(i+ " ping FAILED for device:
"+entry['Hostname']+" at interface: "+entry['Interface']+"\n")

    return logs
```

Next, create the tab on the website for troubleshooting both in the website.py file and a new file that was created called templates/troubleshoot.html

Here is the troubleshoot.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Troubleshoot</title>
    <style>
        body {
            background-color: #121212;
```

```
            color: #ffffff;
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }

        h1, h2, h3 {
            color: #bb86fc;
        }

        button {
            background-color: #bb86fc;
            color: #121212;
            padding: 10px 15px;
            border: none;
            border-radius: 4px;
            cursor: pointer;
            font-size: 16px;
            margin-top: 10px;
        }

        button:hover {
            background-color: #9a67d9;
        }

        pre {
            background-color: #1e1e1e;
            padding: 15px;
            border-radius: 4px;
            overflow-x: auto;
            white-space: pre-wrap;
            margin-top: 10px;
            color: #e0e0e0;
        }

        .header {
            text-align: center;
            padding: 20px;
            background-color: #1e1e1e; /* Header background */
            border-bottom: 2px solid #333; /* Divider line */
        }
    </style>
</head>
<body>
    <div class="header">
```

```
        <h1>Logan's Netman Tool</h1>
    </div>

    <h1>Troubleshoot</h1>
    <form method="POST">
        <button type="submit">Troubleshoot</button>
    </form>

    <!-- Output Section -->
    {% if output %}
    <h3 class="my-4">Troubleshooting Results:</h3>
    <pre>{{ output }}</pre>
    {% endif %}
</body>
</html>
```

After these are created, add the tests for the Code coverage in the unit_tests.py file:

```
class routerTests(unittest.TestCase):

    def test_network_ping_test(self):
        self.assertTrue(network_ping_test())
    def test_netconf(self):
        self.assertTrue(netconf_test())
    def test_snmp_cpu(self):
        self.assertTrue(cpu_test())
    def test_snmp_traps(self):
        self.assertTrue(traps_test())
    def test_snmp_syslog(self):
        self.assertTrue(syslog_test())
    def test_create_user_pass(self):
        self.assertTrue(create_user_pass_test())
    def test_update_router_credentials(self):
        self.assertTrue(update_router_credentials_test())
    def test_configure_arista_device(self):
        self.assertTrue(configure_arista_device_test())
    def test_change_passwords(self):
        self.assertTrue(change_passwords_test())
    def test_createAccess(self):
        self.assertTrue(createAccess_test())
    def test_createCore(self):
        self.assertTrue(createCore_test())
    def test_createEdge(self):
        self.assertTrue(createEdge_test())
```

```
    def test_get_neighborships(self):
        self.assertTrue(get_neighborships_test())
    def test_get_route_table(self):
        self.assertTrue(get_route_table_test())
    def test_get_cpu(self):
        self.assertTrue(get_cpu_test())
    def test_get_ip_connectivity(self):
        self.assertTrue(get_ip_connectivity_test())
    def test_sshInfo(self):
        self.assertTrue(sshInfo_test())
    def test_pcap(self):
        self.assertTrue(pcap_test())
    def test_devices(self):
        self.assertTrue(devices_test())


if __name__ == '__main__':
    data = [
            {"name": "SNMP.py", "count": SNMP_count, "total":
count_functions_in_file("/home/student/Documents/CSCI5840_Advanced_Network_Automa
tion/Lab2/SNMP.py")},
            {"name": "NETCONF.py", "count": NETCONF_count, "total":
count_functions_in_file("/home/student/Documents/CSCI5840_Advanced_Network_Automa
tion/Lab2/NETCONF.py")},
            {"name": "passwords.py", "count": passwords_count, "total":
count_functions_in_file("/home/student/Documents/CSCI5840_Advanced_Network_Automa
tion/Lab4/passwords.py")},
            {"name": "playbookCreation.py", "count": playbookCreation_count,
"total":
count_functions_in_file("/home/student/Documents/CSCI5840_Advanced_Network_Automa
tion/Lab4/playbookCreation.py")},
            {"name": "troubleshooting.py", "count": troubleshooting_count,
"total":
count_functions_in_file("/home/student/Documents/CSCI5840_Advanced_Network_Automa
tion/Lab4/troubleshooting.py")}
        ]
    with
open("/home/student/Documents/CSCI5840_Advanced_Network_Automation/Lab7/counts.cs
v", mode="w", newline="") as file:
        writer = csv.DictWriter(file, fieldnames=["name", "count", "total"])
```

Once these are done, simply run a tshark instance to capture any Syslog messages, you can leave this on for as long as you want or run it in the background:

```
tshark -w traps.pcap -i CR_e1-1
```

After that, all you must do is press the troubleshoot button, and it will start to automatically troubleshoot problems in the network.