```
'''
                COSC364 (RIPv2 Routing Protocol)
        Authors: Haider Saeed (msa280), Drogo Shi (msh217)
                    Date: 07/03/2022
                Filename: RIPv2_ConfigureFile.py

Program Definition: Reads the configuration files of and router and prints out
                any error messages. Otherwise, it creates and binds to
                sockets.
'''

import sys
import configparser  # ConfigParser class which implements a basic configuration language
import time
import socket
import select
import threading  # Used timer here so system load will not affect the time.
import random
from RIPv2_Router import*



LOCAL_HOST = '127.0.0.1'


class RIPv2_ConfigureFile():
    ''' Configure class which reads and processes the configuration file of a router
    using filename. It then tries to create and bind to the sockets. '''


    def __init__(self, config_file):
        """ Initiates the configuration file. """

        self.config_file = config_file
        self.router_info = {}
        self.neighbor = {}



    def router_id_check(self, router_id, port_type):
        """ Checks if the router_id is within the supported parameter range. """

        if (router_id == ''):
            exit_msg(f'Failure: {port_type} router ID check failed! No Router ID parameter.')
        elif (1 <= int(router_id) <= 64000):
            return True
        else:
            print_msg(f'Failure: {port_type} router ID check failed.')
            if (int(router_id) < 1):
                exit_msg(f'{port_type} router ID is less than 1.')
            else:
                exit_msg(f'{port_type} router ID is greater than 64000.')




    def cost_check(self, cost):
        """ Checks if the cost is within the supported parameter range. """

        if (cost == ''):
            exit_msg(f'Failure: Output router cost check failed! No cost values.')
        elif (1 <= int(cost) <= 15):
            return True
        else:
            print_msg('Failure: Output router cost check failed! ')
```

```
        if (int(cost) < 1):
            exit_msg('Output router cost is less than 1.')
        else:
            exit_msg('Output router cost is greater than 15.')




def port_check(self, port, port_type):
    """ Checks if the port is within the supported parameter range. """

    if (port == ''):
        exit_msg(f'Failure: {port_type} router port check failed! No port values.')
    elif (1024 <= int(port) <= 64000):
        return True
    else:
        print_msg(f'Failure: {port_type} router port check failed!')
        if (int(port) < 1024):
            exit_msg(f'{port_type} router port value is less than 1024.')
        else:
            exit_msg(f'{port_type} router port value is greater than 64000.')




def get_router_id(self, config):
    """ Gets the router id from the config file of the router after
    performing sanity checks. """

    try:
        router_id = config['Router_Info']['router_id']
    except:
        exit_msg('Failure: Router ID field is missing!')

    if (self.router_id_check(router_id, 'Own')):
        self.router_info['router_id'] = int(router_id)




def get_inputs(self, config):
    """ Gets the router's input ports then check if any of the input ports
    exist in output_ports."""

    input_ports = []

    try:
        router_inputs = config['Router_Info']['input_ports'].split(', ')
    except KeyError:
        exit_msg("Failure: Router's input port field is missing!")

    for input_port in router_inputs:
        self.port_check(input_port, 'Input')

        if (int(input_port) not in input_ports):
            input_ports.append(int(input_port))
        else:
            exit_msg("Failure: Repeated input ports found.")

    return input_ports
```

```
def get_outputs(self, config):
    """ Gets the router's output values."""
    try:
        router_outputs = config['Router_Info']['outputs'].split(', ')
    except:
        exit_msg("Failure: Router's output field is missing!")


    if (router_outputs == ['']):
        exit_msg('Failure: No router output parameters!')

    outputs = []

    # Converting (5001-2-3) to (5001, 2, 3)
    for output in router_outputs:
        params = output.split('-')
        outputs.append(params)

    return outputs




def router_output_format_check(self, output):
    """ Checks if the router outputs exist and return the
    parameter values. """

    port = None
    cost = None
    router_id = None

    try:
        port = output[0]
    except:
        exit_msg('Failure: Router output has no port value.')

    try:
        cost = output[1]
    except:
        exit_msg('Failure: Router output has no cost value.')

    try:
        router_id = output[2]
    except:
        exit_msg('Failure: Router output has no router id value.')

    return (port, cost, router_id)




def read_outputs(self, config):
    """ Gets the output ports of the router. """

    outputs = self.get_outputs(config)
    output_ports = []
    self.router_info['outputs'] = []

    for output in outputs:

        (port, cost, router_id) = self.router_output_format_check(output)

        id_check_passed = self.router_id_check(router_id, 'Output')
        cost_check_passed = self.cost_check(cost)
        port_check_passed = self.port_check(port, 'Output')
        port_not_repeated = int(port) not in output_ports
```

```
        if (id_check_passed and cost_check_passed and port_check_passed and port_not_repeated):
            output_ports.append(port)
            output_format = {'router_id': int(router_id), 'port': int(port), 'cost': int(cost)}
            self.router_info['outputs'].append(output_format)
            self.neighbor[int(router_id)] = [int(cost), int(port)]
        else:
            exit_msg("Failure: Router output check failed! Wrong output values.")


    input_ports = self.get_inputs(config)

    # Checks if any port number from input port exists in output ports
    for port in input_ports:
        if (int(port) in output_ports):
            exit_msg('Failure: No output port numbers can be in input ports.')

    print_msg('Success: Router output parameter checks have passed.')




def create_and_bind_socket(self, port):
    """Creates socket for the given port and attempts to bind to it. """

    # Trying to create a UDP socket for each port.
    try:
        self.router_info['inputs'][port] = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        print_msg('Success - Created socket for Port #' + str(port))
    except socket.error as message:
        exit_msg('Failure - Unable to create socket. ' + str(message))


    # Trying to bind each port to the socket. #try to move this to router class
    try:
        self.router_info['inputs'][port].bind((LOCAL_HOST, port))
        print_msg('Success - Bound Port #' + str(port) + ' to the socket')
    except socket.error as msg:
        exit_msg('Failure - Unable to bind port to socket. ' + str(msg))

    print_msg("Success: Router is bound to all sockets.")




def read_inputs(self, config):
    """ Gets the input ports for the router and for each port opens a UDP
    socket and attempts to bind to it. """

    input_ports = self.get_inputs(config)

    self.router_info['inputs'] = {}
    for port in input_ports:
        self.create_and_bind_socket(port)




def read_and_process_file(self):
    """ Starts processing and reading the configuration file of the router
    while performing sanity checks. """

    try:
        config = configparser.ConfigParser()
        config.read(self.config_file)
    except configparser.ParsingError:
```

```
            exit_msg('Failure: Parameter values are not in a single line.')

        self.get_router_id(config)
        self.read_outputs(config)
        self.read_inputs(config)

        print_msg('Success - All parameters passed the sanity checks')



def print_msg(message):
    """ Prints the message and the time at which it was sent. """
    current_time = time.strftime("%Hh:%Mm:%Ss")
    print("[" + current_time + "]: " + message)


def exit_msg(message):
    """ Prints the message if something goes wrong with starting the router. """
    print_msg(message)
    print_msg('Failure: Router failed to start.')
    sys.exit()
```