C:\Users\uberT\Desktop\cosc364-rip-assignment\Config_file_reader.py

```
import sys

def readConfig(filename):
    file = open(filename, "r")
    lines = file.readlines()
    #stripping the newline character from each line
    for i in range(len(lines)-1): lines[i] = lines[i].strip()

    #most of the get functions return -1 when there is an error
    routerId = getRouterId(lines)
    if(routerId == -1):
        raise Exception("Router ID not specified or specified incorrectly")

    inputPorts = getInputPorts(lines)
    if(inputPorts == -1):
        raise Exception("Input ports not specified or specified incorrectly")

    outputPorts = getOutputPorts(lines, inputPorts)
    if(outputPorts == -1):
        raise Exception("Output ports not specified or specified incorrectly")

    #no error checking on timer values as it defaults to 30-180-120
    timerValues = getTimerValues(lines)

    file.close()
    return (routerId, inputPorts, outputPorts, timerValues)

def getRouterId(lines):
    #iterate through all lines for a valid parameter line
    for line in lines:
        try:
            line = line.split()
            #check if line is a valid router id then return the router id
            if(line[0] == "router-id" and 1 <= int(line[1]) <= 64000 and len(line)
 == 2):
                return int(line[1])
        except:
            #line treated as a comment and ignored
            buffer = 0
    #no properly formatted line found, return -1 as indication of error
    return -1

def getInputPorts(lines):
    #iterate through all lines for a valid parameter line
    for line in lines:
        try:
            line = line.split()
            #check if line is indicated as an input port parameter then check the
            ports are valid and add them to the input ports list
            if(line[0] == "input-ports"):
                inputPortsEstablished = True
```

```
                        inputPorts = []
                        for i in line[1:]:
                            if(1024 <= int(i) <= 64000 and int(i) not in inputPorts):
                                inputPorts.append(int(i))
                            else:
                                inputPortsEstablished = False
                        #if all the input ports are valid, return the input ports list
                        if inputPortsEstablished: return inputPorts
                except:
                    #line treated as a comment and ignored
                    buffer = 0
            #no properly formatted line found, return -1 as indication of error
            return -1

        def getOutputPorts(lines, inputPorts):
            #iterate through all lines for a valid parameter line
            for line in lines:
                try:
                    line = line.split()
                    #check if line is indicated as an output port parameter then check ↵
↳                   the ports are valid and add them to the output ports list
                    if(line[0] == "outputs"):
                        outputPortsEstablished = True
                        outputPorts = []
                        for i in line[1:]:
                            outputPort = []
                            i = i.split("-")
                            if(len(i) == 3 and int(i[0]) not in inputPorts and 1024 <= int↵
↳                           (i[0]) <= 64000 and 1 <= int(i[1]) <= 15 and 1 <= int(i[2]) <=↵
↳                            64000):
                                #have to iterate through outputPorts to check i isn't in ↵
↳                               it since it is a 2D list
                                for j in outputPorts:
                                    if(int(i[0]) == j[0]):
                                        outputPortsEstablished = False
                                outputPort.append(int(i[0]))
                                outputPort.append(int(i[1]))
                                outputPort.append(int(i[2]))
                                outputPorts.append(outputPort)
                            else:
                                outputPortsEstablished = False
                        #if all the output ports are valid, return the output ports list
                        if outputPortsEstablished: return outputPorts
                except:
                    #line treated as a comment and ignored
                    buffer = 0
            #no properly formatted line found, return -1 as indication of error
            return -1

        def getTimerValues(lines):
            #iterate through all lines for a valid parameter line
```

```
            for line in lines:
                try:
                    line = line.split()
                    #check if line is indicated as a timer values parameter then return the timer values list if the timer values are valid
                    if(line[0] == "timer-values"):
                        timerValues = line[1].split("-")
                        timerValues[0] = int(timerValues[0])
                        timerValues[1] = int(timerValues[1])
                        timerValues[2] = int(timerValues[2])
                        #check timer ratios are correct
                        if(timerValues[1] / timerValues[0] == 6 and timerValues[2] / timerValues[0] == 4 and len(timerValues) == 3):
                            return timerValues
                except:
                    #line treated as a comment and ignored
                    buffer = 0
            #no properly formatted line found, return the default values
            print("Timer values not specified or specified incorrectly, setting to default")
            return [30, 180, 120]


        def main():

            routerId, inputPorts, outputPorts, timerValues = readConfig(sys.argv[1])
            print("Router ID:", routerId)
            print("Input Ports:", end=" ")
            for i in inputPorts: print(i, end=", ")
            print()
            print("Output Ports:", end=" ")
            for i in outputPorts: print(i, end=", ")
            print()
            print("Timer Values:", timerValues)

            inputSockets = createSockets(inputPorts)
            updateTimer, timeoutTimer, garbageTimer = timerValues
            incomingQueue = []

            while True:
                read, write, special = select.select(inputSockets, [], [])

                for i in read:
                    try:
                        addr, data = i.recvfrom(BUFFER_SIZE)
                        incomingQueue.append((addr, data))
                        print(incomingQueue.pop(0))
                    except:
                        print("Error recieving a packet")

                if need_periodic_update(next_update, updateTimer):
                    # send update
```

```
                    print("periodic update time")



        if __name__ == '__main__':
            main()
```