```
    ############## Application #2 - Part #1 ##############


import paramiko

import threading

import os.path

import subprocess

import time

import sys

import re



#Checking IP address file and content validity

def ip_is_valid():

    check = False

    global ip_list


    while True:

        #Prompting user for input

        print "\n# # # # # # # # # # # # # # # # # # # # # # # # # # # # #\n"

        ip_file = raw_input("# Enter IP file name and extension: ")

        print "\n# # # # # # # # # # # # # # # # # # # # # # # # # # # # #"


        #Changing exception message

        try:

            #Open user selected file for reading (IP addresses file)
```

```
        selected_ip_file = open(ip_file, 'r')

        #Starting from the beginning of the file
        selected_ip_file.seek(0)

        #Reading each line (IP address) in the file
        ip_list = selected_ip_file.readlines()

        #Closing the file
        selected_ip_file.close()

    except IOError:
        print "\n* File %s does not exist! Please check and try again!\n" % ip_file

    #Checking octets
    for ip in ip_list:
        a = ip.split('.')

        if (len(a) == 4) and (1 <= int(a[0]) <= 223) and (int(a[0]) != 127) and (int(a[0]) != 169 or int(a[1]) !=
254) and (0 <= int(a[1]) <= 255 and 0 <= int(a[2]) <= 255 and 0 <= int(a[3]) <= 255):
            check = True
            break

        else:
            print '\n* There was an INVALID IP address! Please check and try again!\n'
            check = False
```

```
            continue


    #Evaluating the 'check' flag
    if check == False:
        continue


    elif check == True:
        break


############## Application #2 - Part #2 ##############


#Checking IP reachability
print "\n* Checking IP reachability. Please wait...\n"


check2 = False


while True:
    for ip in ip_list:
        ping_reply = subprocess.call(['ping', '-c', '2', '-w', '2', '-q', '-n', ip])


        if ping_reply == 0:
            check2 = True
            continue


        elif ping_reply == 2:
```

```
            print "\n* No response from device %s." % ip
            check2 = False
            break

        else:
            print "\n* Ping to the following device has FAILED:", ip
            check2 = False
            break

    #Evaluating the 'check' flag
    if check2 == False:
        print "* Please re-check IP address list or device.\n"
        ip_is_valid()

    elif check2 == True:
        print '\n* All devices are reachable. Waiting for username/password file...\n'
        break

#Checking user file validity
def user_is_valid():
    global user_file

    while True:
        print "# # # # # # # # # # # # # # # # # # # # # # # # # # # # #\n"
        user_file = raw_input("# Enter user/pass file name and extension: ")
```

```
        print "\n# # # # # # # # # # # # # # # # # # # # # # # # # # # # #"

        #Changing output messages
        if os.path.isfile(user_file) == True:
            print "\n* Username/password file has been validated. Waiting for command file...\n"
            break

        else:
            print "\n* File %s does not exist! Please check and try again!\n" % user_file
            continue

#Checking command file validity
def cmd_is_valid():
    global cmd_file

    while True:
        print "\n\n# # # # # # # # # # # # # # # # # # # # # # # # # # # # #\n"
        cmd_file = raw_input("# Enter command file name and extension: ")
        print "\n# # # # # # # # # # # # # # # # # # # # # # # # # # # # #"

        #Changing output messages
        if os.path.isfile(cmd_file) == True:
            print "\n* Sending command(s) to device(s)...\n"
            break
```

```
        else:
            print "\n* File %s does not exist! Please check and try again!\n" % cmd_file
            continue

#Change exception message
try:
    #Calling IP validity function
    ip_is_valid()

except KeyboardInterrupt:
    print "\n\n* Program aborted by user. Exiting...\n"
    sys.exit()

#Change exception message
try:
    #Calling user file validity function
    user_is_valid()

except KeyboardInterrupt:
    print "\n\n* Program aborted by user. Exiting...\n"
    sys.exit()

#Change exception message
try:
    #Calling command file validity function
```

```
    cmd_is_valid()


except KeyboardInterrupt:
    print "\n\n* Program aborted by user. Exiting...\n"
    sys.exit()


    ############## Application #2 - Part #3 ##############


#Open SSHv2 connection to devices
def open_ssh_conn(ip):
    #Change exception message
    try:
        #Define SSH parameters
        selected_user_file = open(user_file, 'r')


        #Starting from the beginning of the file
        selected_user_file.seek(0)


        #Reading the username from the file
        username = selected_user_file.readlines()[0].split(',')[0]


        #Starting from the beginning of the file
        selected_user_file.seek(0)


        #Reading the password from the file
```

```
password = selected_user_file.readlines()[0].split(',')[1].rstrip("\n")

#Logging into device
session = paramiko.SSHClient()

#For testing purposes, this allows auto-accepting unknown host keys
#Do not use in production! The default would be RejectPolicy
session.set_missing_host_key_policy(paramiko.AutoAddPolicy())

#Connect to the device using username and password
session.connect(ip, username = username, password = password)

#Start an interactive shell session on the router
connection = session.invoke_shell()

#Setting terminal length for entire output - disable pagination
connection.send("terminal length 0\n")
time.sleep(1)

#Entering global config mode
connection.send("\n")
connection.send("configure terminal\n")
time.sleep(1)

#Open user selected file for reading
```

```
selected_cmd_file = open(cmd_file, 'r')

#Starting from the beginning of the file
selected_cmd_file.seek(0)

#Writing each line in the file to the device
for each_line in selected_cmd_file.readlines():
    connection.send(each_line + '\n')
    time.sleep(2)

#Closing the user file
selected_user_file.close()

#Closing the command file
selected_cmd_file.close()

#Checking command output for IOS syntax errors
router_output = connection.recv(65535)

if re.search(r"% Invalid input detected at", router_output):
    print "* There was at least one IOS syntax error on device %s" % ip

else:
    print "\nDONE for device %s" % ip
```

```
        #Test for reading command output
        #print router_output + "\n"

        #Closing the connection
        session.close()

    except paramiko.AuthenticationException:
        print "* Invalid username or password. \n* Please check the username/password file or the device configuration!"
        print "* Closing program...\n"

    ############## Application #2 - Part #4 ##############

#Creating threads
def create_threads():
    threads = []
    for ip in ip_list:
        th = threading.Thread(target = open_ssh_conn, args = (ip,))   #args is a tuple with a single element
        th.start()
        threads.append(th)

    for th in threads:
        th.join()
#Calling threads creation function
create_threads()
#End of program
```