Using Pexpect to establish a SSH connection

Step 1

Configure router r1 to support SSHv2 only.

```
r1>en
Password: cisco
r1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
r1(config)#ip domain-name cisco.com
r1(config)#crypto key generate rsa
The name for the keys will be r1.cisco.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys.  Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 0 seconds)

r1(config)#
* Apr 12 13:52:35.463: %SSH-5-ENABLED: SSH 1.99 has been enabled
r1(config)#ip ssh version 2
r1(config)#ip ssh rsa keypair-name r1.cisco.com
r1(config)#
* Apr 12 13:54:11.919: %SSH-5-DISABLED: SSH 2.0 has been disabled
* Apr 12 13:52:41.337: %SSH-5-ENABLED: SSH 2.0 has been enabled
r1(config)# end
r1#
```

Step 2

Establish an SSH connection to router r1. Your application should include error checking.

- IP Address: 10.30.30.1
- Username: cisco
- Password: cisco
- Enable Password: cisco

```
import pexpect

ip_address = '10.30.30.1'
username = 'cisco'
password = 'cisco'
password_enable = 'cisco'

#create the pexpect session
session = pexpect.spawn('ssh ' + username + '@' + ip_address, timeout=20)
result = session.expect(['Password:', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print '--- FAILURE! creating session for: ', ip_address
    exit()

# Session expecting password, enter it here
session.sendline(password)
result = session.expect(['>', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print ' FAILURE! entering password: ', password
    exit()
```

Note

If you are unable to connect to the router via your script, it is recommended that you attempt to `ssh` from the command line. If you have a host key validation error, it is advisable to remove the offending RSA key as per the instructions outlined in the output from the SSH command.

Step 3

Set the hostname to R1.

```
# Enter enable mode
session.sendline('enable')
result = session.expect(['Password:', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print ' Failure! entering enable mode'
    exit()

# Send enable password
session.sendline(password_enable)
result = session.expect(['#', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print ' Failure! entering enable mode after sending password'
    exit()

# Issue config t command.
session.sendline('config t')
result = session.expect(['r1\(config\)#', pexpect.TIMEOUT])

# Check for error, if so then print error and exit
if result != 0:
    print ' Failure! entering config mode'
    exit()

# Change the hostname to R1
session.sendline('hostname R1')
result = session.expect(['R1\(config\)#', pexpect.TIMEOUT])
```

Step 4

Verify that the hostname is correctly set, print a success message and close the SSH connection. If the hostname is not correctly set, print a failure message.

Answer

To verify that the hostname has been changed, check that the prompt has changed to 'R1(config)#'. Before exiting, your application should exit out of configuration mode and enable mode.

```
# Check for error, do not exit - exit config mode first
if result != 0:
    print ' Failure! setting hostname'

# Exit config mode and exit enable mode

session.sendline('exit')
session.sendline('exit')


print '--- Success! connecting to: ', ip_address
print '---               Username: ', username
print '---               Password: ', password
print '------------------------------------------------------\n'

# End session

session.close()
```