TITLE
Project 1: Recursive DNS client and DNS servers

DUE
Wed 04 Mar 2020 14:00

NUMBER OF RESUBMISSIONS ALLOWED
Unlimited

ACCEPT RESUBMISSION UNTIL
Wed 04 Mar 2020 14:00

Project 1: Recursive DNS client and DNS servers

OVERVIEW

The goal of the project is to implement
a **simplified DNS system** consisting of a **client program** and **two server programs**:

RS (a simplified root DNS server)
TS (a simplified top-level DNS server).

In **project 0** (your first HW),
you have already seen a client-server program with **one socket** in the client and the server.

In **this project**, you will **extend that implementation** to have **two sockets** in the client program.

One socket will be used to **communicate with RS** and the other **with TS**.

The **RS** and **TS** programs each maintain a `DNS_table` consisting of three fields:

```
hostname
IP address
flag (A or NS)
```

You need to choose the **appropriate data structure** to store the **values** for each **entry**.

The **client** always **connects first to RS,** sending the **queried hostname** as a **string**.

The **RS** program does a **look-up** in its `DNS_table`,
and if there is a **match**, **sends** the **entry** as a **string**:

```
queried_hostname its_ip_address A
```

If there is **no match**, **RS sends** the **string**:

```
ts_hostname – NS
```

where `ts_hostname` is the **name of the machine** on which the **TS** program is **running**.

If the **client** receives a **string** with `"A"`, it **outputs** the **received string as is**.

On the other hand, if the **client** receives a **string** with `"NS"`,
it uses the `ts_hostname` portion of the **received string** to determine the **IP address**
of the **machine running** the **TS** program and **connects** to the **TS** program using a **second socket**.

The **client** then **sends** the **queried hostname** as a **string** to **TS**.

The **TS** program does a **look-up** on the **hostname** in its `DNS_table`,
and if there is a **match —** it **sends** the **entry,** as a **string,** to the **client:**

```
queried_hostname its_ip_address A
```

Otherwise, it sends an **error string**:

```
queried_hostname – error: HOST NOT FOUND
```

`'queried_hostname'` could range from `localhost`, to `foo.bar.rutgers.edu`, etc.

The **client outputs** the **string received** from **TS as is**.

Note that all DNS lookups are case-insensitive.

If there is a **hit** in the local `DNS_table`,
the **server programs must respond**
with the **version of the string that is in their local** `DNS_table`.

PROJECT TESTING

As part of your submission, you will submit three sources/files:

```
rs.py
ts.py
client.py
README
```

We will be **running** the three **programs** (`*.py`) on the **iLab machines with Python 2.7**.
Please do not assume that all programs will run on the same machine or that all connections are made to the `localhost`.

We reserve the right to test your programs with **local** and **remote socket connections**, with `client.py`, `ts.py`, and `rs.py` **each running on independent machines**, for example.

You are welcome to simplify the initial development and debugging of your project, and get off the ground by running all programs on one machine first.

However, you must eventually ensure that the programs can work across multiple machines.

The programs **must work** with the following **command line arguments**:

```
python ts.py ts_listen_port
      e.g. python ts.py 8345

python rs.py rs_listen_port
      e.g. python rs.py 8345

python client.py rs_hostname rs_listen_port ts_listen_port
      e.g. python client.py pwd.cs.rutgers.edu 8345 50007
```

`ts_listen_port` is the **port** on which **TS** listens for **requests**;
`rs_listen_port` is the **port** on which **RS** listens for **requests**;
`rs_hostname` is the **hostname** of the **machine running** the **RS** program.

The **hostname strings** to be **queried** will be **given one per line in a file**:

```
PROJI-HNS.txt
```

The **entries** of the local `DNS_table`s, **one each** for **RS** and **TS**, will be **strings** with **fields separated by whitespace**. There will be **one entry per line**. You can see the **format** in both

```
PROJI-DNSRS.txt
PROJI-DNSTS.txt
```

Your **server** programs should **populate the `DNS_table`** by **reading** the **entries** from the **corresponding files**. Your **client** program should **output** the **results to a file named**

```
RESOLVED.txt
```

with **one line per result**.

We will test your programs by running them with the hostnames and tables in the attached input files (`*.txt`) as well as with new hostnames and table configurations. You will be graded based on the outputs in `RESOLVED.txt`. **Your programs should not crash on correct inputs.**

README file

In addition to your programs, you must also submit a `README` file with clearly delineated sections for the following:

0. Please write down the full names and NetIDs for both of your team members.

1. Briefly discuss how you implemented your recursive client functionality.

2. Are there known issues or functions that are currently not working in your attached code? If so, explain.

3. What problems did you face developing code for this project?

4. What did you learn by working on this project?

SUBMISSION

Turn in your project on Sakai assignments. **Only one team member needs to submit**. Please upload a **single** `.zip` file consisting of

```
client.py
rs.py
ts.py
README
```

TIPS

Run your programs in the following order:

0. `ts.py`
1. `rs.py`
2. `client.py`

DNS lookups are case-insensitive.

It is okay to assume that each DNS entry or host name is smaller than 200 characters.

START EARLY to allow plenty of time for questions on Piazza should you run into difficulties.