Project milestone 2 testing strategies

As you work to develop your web server, you are going to need to send it various types of input (good, malformed, etc) to ensure that it is working correctly. The purpose of this document is to give you some practical strategies for doing so.

Using Pure Command Line Tools

There are several commands that might be useful for you to use

1. **netcat**

With -l flag, you are creating a server socket, listening on the "localhost" address. You bind to a particular port via the -p flag.

```
$ nc -l -p 8885
```

This will listen for connections on port 8885.

Try it:
Start a server session using "nc -l -p xxxx", and then point a web browser to that host/port and see what it prints out. What do you see?

In this version, you create connection to a server (and port) as indicated

```
$ nc google.com 80
```

After you create the connection, you would be able to type in the message/request you want to send to the server. Note that if you use your keyboard as input, when you hit the "enter" or "return" key, it isn't clear what character(s) will be sent to the server. Some platforms send a carriage return (\r), some send a new line (\n), and some send both a carriage return and a newline (\r\n). It depends on whether you're on Linux, Mac, or Windows (or perhaps something else?)

We're going to want more control over **exactly** what gets sent to the server. Furthermore, we're going to want to build up a set of *test vectors*, or testing inputs, that we can send to the server. Rather than typing them out each time, we'd like to be able to read and write them to a file.

However, writing the request message each time would be tedious. So we can use:

2. **printf & pipe**

printf works just like the printf you might have used in your C/C++ program.

```
$ printf 'GET /index.html HTTP/1.1\r\nHost: MyHost\r\n\r\n'
```

Try it:
Type the above command into Linux and see what it produces. Now you have control over exactly whether you send a \r or a \n or a \r\n.

With the string literal get printed, you can pipe it into a file with '>'
printf 'GET /index.html HTTP/1.1\r\nHost: MyHost\r\n\r\n' > request.input

After that, you would have a file that contains the request content you want to send to the server, and you don't need to type the content again and again.

Putting it together:
We can compose the above two steps so that we can generate a possible input

```
$ printf 'GET /index.html HTTP/1.1\r\nHost: MyHost\r\n\r\n' > request.input
```

Then we can send that input to your webserver via:

```
$ cat request.input | nc localhost 8885
```

You can even store the output of your server into a file as well!

```
$ cat request.input | nc localhost 8885 > response.output
```

1. **Hexdump**

You can dig into the content of the file with hexdump. This part is optional but make sure the content you want to send is correct!

This gives you the hexdecimal form of the content in your file

```
$ hexdump [filename]
```

This gives you character form of the content in your file

```
$ hexdump -c [filename]
```

Try it!

Combining the tools for testing

1. Create a test input file that contains the request message

```
$ printf 'GET /index.html HTTP/1.1\r\nHost: MyHost\r\n\r\n' > request.input
```

2. Verify the content is what you want to send

```
$ hexdump -c request.input
```

3. Use nc to send the request and see what's the response from the server

```
$ cat request.input | nc -l 8885
```

Bravo

Using Python

```
from socket import socket

# Create connection to the server
s = socket()
s.connect(("localhost", 8885));

# Compose the message/HTTP request we want to send to the server
msgPart1 = b"GET /index.html HTTP/1.1\r\nHost: Ha\r\n\r\n"

# Send out the request
s.sendall(msgPart1)

# Listen for response and print it out
print (s.recv(4096))

s.close()
```