Message Format

In a DNS request, the application data has the following format in network byte order:

```
0              1                  2                3              4

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|      Message Type (1)                |      Return Code (0)             |
|                         Message Identifier(e.g. 5)                      |
|     Operation Length (e.g 23)        |      Result Length (0)           |
//                        Operation String                              //

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In a DNS response, the application data has the following format in network byte order:

```
0              1                2                  3               4

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

|     Message Type (2)              |        Return Code (0 or 1)         |
|                        Message Identifier(e.g. 5)                        |
|     Operation Length (e.g 23)     |         Result Length (41)          |
|                         Operation Section                               //
//                                 |
|                        Result Section                                    //
//                                                                        |

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Message Semantics

Message Type (16bit):1 on request, 2 on response

Return Code (16bit): 0 on request. For response, 0 if operation is successful. 1 if light bulb is not functional (light bulb is off). 2 if color is not support (will listed what color support in below). 3 if operation string is incorrect. 4 if light bulb singal is broken which unable to complete the operation. (Will specific more details below)

Message Identifier (32bit): Uniquely identifies an operation in request. Server will echo same number back in response. Randomly generate between 1 and 100.

Operation Length (16 bit): In request and response, the length of resource record in Operation Section in bytes.

Result Length (16 bit): 0 in result (No result section in response). In response, length of resource record in Result section in bytes.

Operation Section (variable Length): In request, String carry the operation in the form of a DNS resource record containing the device, in this case "lightbulb" and the choice of operation to be perform on the device. The choice of operations is turn on the light bulb, which is "operation1", followed by the color you want the light bulb on that separate by a space, e.g. "red". For second choice, "operation2", is to change the color of the light bulb, which again separate by a space after specific the operation number, e.g. "blue". For third choice is to determine the status of the light is on or off, and the color of the light bulb if the light bulb is on, which can be use using "operation3". Last operation is turning the light off, which specific by "operation4". Echoed back in server response. Below is the example possible operation and the format to be used for operation section.

Example:

- "lightbulb.operation 1 red"
- "lightbulb.operation 2 blue"
- "lightbulb.operation 3"

Note:

- Possible color choice: Red, Blue, Green, Yellow, White, Pink, Purple, Orange. (Recommend use Array to contain the choice)

Result Section (variable Length): In request, there is no result section. In response, the server will includes the device ("lightbulb") follow by the operation number that the user request, followed by the result include after the operation if successful. Example: of operation1 is successful, then it will include light bulb is on, and the color the user change to. All result string will include “SUCCESS” when operation complete with no error. If any error that cause the operation is unsuccessful, result section will include the string that specific the device follows by the operation number with the string “FAIL”. Below is the possible result example and more specific details:

Example:

For operation 1:

- “lightbulb.operation 1 ON red SUCCESS”
- “lightbulb.operation 1 FAIL”

For operation 2:

- “lightbulb.operation 2 CHANGE blue SUCCESS”
- “lightbulb.operation 2 FAIL”

For operation 3:

- “lightbulb.operation 3 details: ON blue”
- “lightbulb.operation 3 details: OFF”

For operation 4:

- “lightbulb.operation 4 OFF SUCCESS”
- “lightbulb.operation 4 FAIL”

If String format is incorrect:

- Include the string from the request, followed by “FAIL” separate by a space.
- Note:
- Return this error if it operation 3 and 4 since no color should specific in these operation. (Incorrect number of argument!)

Example:

- “lightorb.op 3 FAIL”
- “lightbulb.operation 3 blue FAIL”

Error code details:

1: Light bulb is Off and try to perform color change. (operation 2)

2: Color is not supported in light bulb. (operation 1 and 2)

3: Incorrect String format (All operation)

4. Light bulb signal is broken

- To implement a broken light bulb, use random number to determine if it is broken.

- Use number between 1 to 100, and even the number is less than or equal to 5, it is broken. (Please do not use the same number form Message Identifier.

Test Output:

1. **Test Case 1: Successful Operation, 4 total case, 1 for each operation:**

- **Operation 1:**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 1 red

Sending Request to 127.0.0.1, 12000:

Message ID: 63

Operation Length: 25 bytes

Result Length: 0 bytes

Operation: lightbulb.operation 1 red


Received Response from 127.0.0.1 12000:

Return Code: 0

Message ID: 63

Operation Length: 25 bytes

Result Length: 36 bytes

Operation: lightbulb.operation 1 red

Result: lightbulb.operation 1 ON red SUCCESS
```

- **Operation 2:**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 2 blue
Sending Request to 127.0.0.1, 12000:
Message ID: 19
Operation Length: 26 bytes
Result Length: 0 bytes
Operation: lightbulb.operation 2 blue

Received Response from 127.0.0.1 12000:
Return Code: 0
Message ID: 19
Operation Length: 26 bytes
Result Length: 41 bytes
Operation: lightbulb.operation 2 blue
Result: lightbulb.operation 1 CHANGE blue SUCCESS
```

- **Operation 3:**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 3
Sending Request to 127.0.0.1, 12000:
Message ID: 81
Operation Length: 21 bytes
Result Length: 0 bytes
Operation: lightbulb.operation 3
```

```
Received Response from 127.0.0.1 12000:
Return Code: 0
Message ID: 81
Operation Length: 21 bytes
Result Length: 39 bytes
Operation: lightbulb.operation 3
Result: lightbulb.operation 3 details: ON GREEN
```

- **Operation 4:**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 4
Sending Request to 127.0.0.1, 12000:
Message ID: 7
Operation Length: 21 bytes
Result Length: 0 bytes
Operation: lightbulb.operation 3

Received Response from 127.0.0.1 12000:
Return Code: 0
Message ID: 7
Operation Length: 21 bytes
Result Length: 33 bytes
Operation: lightbulb.operation 3
Result: lightbulb.operation 4 OFF SUCCESS
```

2. **Test Case 2: Client Output (Unsuccessful Operation):**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 2 Neon
Sending Request to 127.0.0.1, 12000:
Message ID: 88
Operation Length: 26 bytes
Result Length: 0 bytes
Operation: lightbulb.operation 2 Neon

Received Response from 127.0.0.1 12000:
Return Code: 2
Message ID: 88
Operation Length: 25 bytes
Result Length: 26 bytes
Operation: lightbulb.operation 2 Neon
Result: lightbulb.operation 2 FAIL
```

- **Incorrect String format case:**

```
$ python3 light-client.py 127.0.0.1 12000 lightorb.op 3
Sending Request to 127.0.0.1, 12000:
Message ID: 21
Operation Length: 13 bytes
Result Length: 0 bytes
Operation: lightorb.op 3
```

```
Received Response from 127.0.0.1 12000:

Return Code: 3

Message ID: 21

Operation Length: 25 bytes

Result Length: 18 bytes

Operation: lightorb.op 3

Result: lightorb.op 3 FAIL
```

3. **Test Case 3: Client Output (Server Not responding, Comment out for the sending part so it won't crash the program if you on Window OS):**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 1 red

Sending Request to 127.0.0.1, 12000:

Message ID: 12

Operation Length: 25 bytes

Result Length: 0 bytes

Operation: lightbulb.operation 1 red

Operation timed out …

Sending Request to 127.0.0.1, 12000:

Operation timed out …

Sending Request to 127.0.0.1, 12000:

Operation timed out … Exiting Program
```

4. **Test Case 4: Lightbulb malfunction:**

```
$ python3 light-client.py 127.0.0.1 12000 lightbulb.operation 1 red

Sending Request to 127.0.0.1, 12000:
```

```
Message ID: 32

Operation Length: 25 bytes

Result Length: 0 bytes

Operation: lightbulb.operation 1 red


Received Response from 127.0.0.1 12000:

Return Code: 4

Message ID: 32

Operation Length: 25 bytes

Result Length: 26 bytes

Operation: lightbulb.operation 1 red

Result: lightbulb.operation 1 FAIL
```

- **In this case the number it got from random is 2, which is less than or equal to 5. Don't forget to include a variable that contain a number between 1 to 100 and if it below or equal to 5, return error code 4. Don't use Message ID!**

Important Note:

The client will perform the following:

1. Read the argument. (Minimum is 4, maximum is 5 when specific operation 1 or 2, which include the color that you want)
   a. IP address of server (127.0.0.1)
   b. Port of server (e.g. 12000)
   c. lightbulb.operation (must use this format, otherwise it will be error code 3)
   d. Number of the operation (1 2 3 4)
   e. Color of the light bulb, only for operation 1 and 2 (error if it operation 3 and 4, error code 3)
2. Send the request using the string format above to the server.
3. Wait for 1 second timeout period. Resend the request up to 3 times if it timeout, otherwise, print out all the details shown in this document.

The server will perform the following:

1. Read 2 argument
   a. IP address of server (127.0.0.1)
   b. Port of server (e.g. 12000)
2. Use **switch** case to identify what operation the request is requesting
- Hint: use default case to handle string that is in incorrect format. Error 3.
3. Respond to DNS request and return all the information specific above.
4. Make sure store all possible color in array. Refer to operation section above for all possible color.

Good Luck!