Exercise: decode network trace

The file `mcs/priv/ping.pcap` in the repository is a PCAP packet trace composed of two packets: one ICMP echo request and one ICMP echo reply, generated by `ping -c1 localhost`.

References for the file and packet formats:

- [PCAP format]
- [IPv4 format]
- [ICMP format]

To verify your understanding of the binary format, you can use [wireshark] or [tshark].

Exercise: decode network trace, part 1

Write function `pcap:render_file(FileName)` to print the decoding like tshark:

```
$ tshark -r ping.pcap
1 127.0.0.1 -> 127.0.0.1 ICMP Echo request id=0xae4b, seq=0/0, ttl=64
2 127.0.0.1 -> 127.0.0.1 ICMP Echo reply   id=0xae4b, seq=0/0, ttl=64
```

To keep your code modular, readable and testable, you will have to write many small functions.

The idea is to keep the parsing completely decoupled from the rendering, so that one could have one parser that allows multiple renderers (text, GUI, web, ...).

Exercise: decode network trace, part 2

Add more information, inspired by the output of `tshark -V`.

If interested in keeping backward compatibility, one could add function `pcap:render_file(FileName, Options)` and reimplement existing API:

```
render_file(FileName) -> render_file(FileName, []).
```

and document that the equivalent of `tshark -V` is obtained by calling `pcap:render_file(FileName, ['V'])`.

(cont.)

Exercise: decode network trace, part 2

```
$ tshark -V -r ping.pcap
Null/Loopback
    Family: IP (2)
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
    Total Length: 84
    Identification: 0x5952 (22866)
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (1)
    Header checksum: 0x0000
    Source: 127.0.0.1
    Destination: 127.0.0.1
Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    ...
```