1. Cryptocurrency Ticker Receiver Library in C++

Description
In trading, a "ticker" is a report of the prices for a specific product, updated continuously throughout the trading session by the various exchanges. A ticker includes information about the traded volume of the asset, as well as the highest buying and the lowest selling price.

In the context of this assignment, the ticker data you'll be fetching will include information about the real-time prices of the cryptocurrency Ethereum (ETH) paired with the stablecoin US Dollar Tether (USDT).

Please develop a real-time cryptocurrency ticker receiver library in C++. The program should fetch this ETH/USDT ticker information from Kraken exchange.
You don't need to construct the json format yourself, instead you can hardcode the endpoint and payload.

P.S.: Websocket endpoint of the Kraken and the corresponding payload for subscribing ticker are listed in the appendix. You can use some command line tools, like `websocat`, to do a subscription test by connecting to the endpoint and sending the payload.

Key Requirements:

- Design a library that allows users to subscribe and receive the ticker data from the different exchanges.
    - The ticker structure should be passed to users as defined on the appendix.
    - The library must be capable of obtaining real-time ticker information for the ETH/USDT currency pair from Kraken.
    - Besides Kraken, you should consider the extendibility of the library for multiple exchanges.

- Please provide listed materials.
    - library source code
    - a main function to demonstrate how to interact with the library
    - build scripts

2. Explain how you would approach identifying performance bottlenecks in a library. Consider a library that you've recently worked with that exhibited slower performance than expected.

Appendix

Ticker Structure:
For this assignment, you should consider the following simple structure for a ticker:

```
struct Ticker {
    double best_ask_price;
    double best_bid_price;
    double close_price;
    double today_high_price;
    double today_low_price;
    double today_open_price;
}
```

API/Websocket Endpoint:

- Kraken: wss://ws.kraken.com

Subscription payload format for the Kraken exchange:

```
{"event": "subscribe", "pair": ["ETH/USDT"], "subscription": {"name":
"ticker"} }
```

Example response format from the Kraken exchange:

```
{"channelID":2276,"channelName":"ticker","event":"subscriptionStatus","p
air":"ETH/USDT","status":"subscribed","subscription":{"name":"ticker"}}
[2276,{"a":["1851.79000",1,"1.00000000"],"b":["1851.51000",1,"1.00000000
"],"c":["1851.86000","0.89542000"],"v":["107.07841802","1828.67105949"],
"p":["1836.55404","1906.18792"],"t":[182,1998],"l":["1827.57000","1827.5
7000"],"h":["1851.86000","1956.55000"],"o":["1847.51000","1913.58000"]},
"ticker","ETH/USDT"]
```

For response format definition, please refer to:
https://docs.kraken.com/websockets/#message-ticker