**Question 1 – Theory**

* 1. Ethereum uses a concept called gas to pay for the execution (computational cost) of smart contracts (the Ethereum programs). Gas is the unit of work expended per computational operation in the Ethereum Virtual Machine. The total cost of an Ethereum transaction is the amount of necessary gas multiplied by the price in GWei per gas unit. GWei varies according to network congestion and user’s preference for a speedy confirmation. Large influences on the transaction fees of Ethereum are the number of pending transactions and the number of available miners. The fewer miners available, the higher the costs and the more pending transactions, the higher the costs due to increased demand and decreased supply. The fees are also dependent on the type of operation the miner is performing, each operation has a set gas cost.
  2. Performing operations on Ethereum network costs gas. For this reason, if traders are planning on developing applications or new tokens on the network, they will need to take into account transaction fees. If you are planning on buying Ether as a store of wealth (in other words, just investing in Ethereum), transaction fees have very little impact on traders.
  3. Automatic market makers (AMMs) are a decentralized exchange protocol that relies on a mathematical formula to price assets. Traditional market making works to get a good price and tight bid-ask spread utilizing an order book exchange. In comparison, AMMs decentralize this process. There is no need for counterparties under the use of AMMs when compared to regular exchanges which require counterparties to execute the transaction.

1. A stable coin is a type of cryptocurrency whose value is pegged to another asset class, for example gold, in order to stabilize the price therefore they are cryptocurrencies with fixed values. Cryptocurrencies are traditionally highly volatile. In order to stabilize the crypto ecosystem, stabile coins were introduced. They bring in the benefits of cryptocurrency along with more price stability. The first type of stable coin is a fiat-collateralised stable coin which are backed by fiat currencies such as the US dollar. The price of these stable coins fluctuates according to the fiat currency. Crypto-backed stable coins are the second way to construct a stable coin. These are predominantly backed by decentralized crypto currencies or assets such as Bitcoin or Ethereum. These are more volatile due to the fact that crypto currencies are highly volatile. Set protocols are utilized to ensure the price of stable coin remains stable and therefore higher collateralization is required. Commodity-backed stable coins are pegged to tangible assets such as precious metals like gold. They are dependent on reserve holdings and value of the pegged asset. Finally, stable coins can be constructed without collateral. These are known as algo-based stable coins. They rely on smart contracts and use algorithms to adjust the supply of stable coins in order to maintain a stable market value. These algorithms respond to market movements to ensure it does not get manipulated.
2. A perpetual future is a special type of futures contract without an expiry date. In other words, one can hold a perpetual future for as long as one would like. A funding rate is a mechanism used to ensure that future prices and index prices converge regularly. When the perpetual futures contract is trading at a premium (i.e. higher than spot market), long positions have to pay shorts due to a positive funding rate. in comparison, short positions pay long positions when the futures price is below the index price. Funding fees are paid between holders of futures and not by the exchange. An arbitrage opportunity therefore arises in holding a short position in the perpetual futures market and purchase the same amount in the spot market which then hedges the total investment. Therefore, one has a market-neutral position and will receive the funding rates with the short position in the perpetual futures market.

**Question 3 – Discussion**

1. BTC/USD: Sharpe ratio = 0, 02…

ETH/USD: Sharpe ratio = 0,04…

The higher the Sharpe ratio, the better the risk-adjusted-performance of the asset, therefore, the ETH/USD has a better risk-adjusted-performance in the year 1 July 2020 – 1 July 2021 for 5-minute returns.

BTC/USD: Sortino ratio = 0,02…

ETH/USD: Sortino ratio = -0.1131245…

A higher Sortino ratio is better as it shows a better performance taking into account only downside risk, therefore BTC performs better than ETH in the year 1 July 2020 – 1 July 2021.

1. Information ratio = 0.017…

The information ratio is not very large, but it is positive which implies BTC is performing above ETH as a benchmark.

1. Tracking Error: BTC Spot vs Perp = 7,6378…

ETH Spot vs Perp = 0,4627…

Tracking error is the divergence between the spot price and the price of the perpetual future. The tracking error for BTC is larger than the tracking error for ETH. Therefore, the divergence between spot and future prices is larger for BTC than ETH.

This is the same as for the seven day rolling tracking error which is also larger for BTC than ETH, once again showing that the divergence between spot price and perpetual futures price is larger for BTC than ETH.

**Question 4**

This question is beyond my skillset as I am not a computer programmer and don’t plan on working in programming, so I have answered this question in written form as to how I would code it if I could program.

In order to create a program to interact with the Coinbase Pro sandbox exchange and place orders, firstly, I created two APIs one in production and one in sandbox pro as shown below:

YkHI2OZFf1gJDZSIbOyU5gELZ6Z/ksW1sJX5T9Ou0WAnmkBecPay6DfTwbHm01NBWc03Y5jMxPQFdPYh3yZlQw==

3d1gCAN0YbtXDYWXWKTQ6xHmsk3Rn4SRuBFCMWTY2BlnGiUL0bDr4gH0ubjBoBW3MeHKcuJSvid3wjVcklz6WA==

Next, I would create an environment by creating variables for each of the sandbox API key, sandbox passphrase and sandbox secret as well as the coinbase url.

Following this, I would install the Coinbase Pro Python package. I would then create a function to insert the coinbase authentication to any necessary functions.

I would then create a function to retrieve the bank account id for the account used to make the deposit. Then a second function would be created to use the returned bank account to make a deposit.

This function would be repeated randomly making deposits and purchasing different cryptocurrencies using the above functions.