Sometimes things don’t go the way you expect them to go. For this midterm submission, I was only able to finish Task 1. Tasks 2 and 3 were huge, and due to a failure in my time management, I failed to achieve the desired tasks. I want to apologize since this is not a correct attempt to submit, this submission is only crumbs, but still, I decided to send the little work I completed.

In task 1, I started setting up the canvas to produce the required solution. This process started by building the header file with the correspondent cpp file. I created the Candlestick header file which contains a constructor function with its correspondent data members. The Candlestick class represents a single candlestick in a trading system. A candlestick stores information such as the opening price, highest price, lowest price, and closing price for a specific period. The next step was to make the application of the previous header file. The purpose of this file is to implement the member functions of the Candlestick class defined in the "CandleStick.h" header file. It contains the constructor implementation for the Candlestick class. Finally in "MerkelMain.h" it is declared the definition of the candle stick "candleStick()" and "displayKnownProducts()" which will handle the candlestick stats retrieving the list of known products from the order book respectively.

The actual application of task 1, is performed in "MerkelMain.cpp," this starts by building the "displayKnownProducts()" function is responsible for retrieving and displaying the list of known products (ask and bid) from the order book. It calls the "getKnownProducts()" function of the OrderBook class to obtain a vector of strings representing the known products. Then, it iterates over the products and prints each product name to the console. The next step was to construct the "candleStickData()" function is used to compute and display the candlestick data for a specific product and order type. It prompts the user to enter an order type (ask or bid) and a product. It then retrieves the unique timestamps from the order book by calling the "getEarliestTime()" and "getNextTime()" functions. After that, it iterates over the unique timestamps, and for each timestamp, it retrieves the orders of the specified order type and product by calling the "getOrders()" function of the OrderBook class. It calculates the highest, lowest, open, and close prices for each timestamp by iterating over the retrieved orders. Finally, it creates a Candlestick object for each timestamp with the calculated values and adds it to a vector. The function prints the candlestick data by iterating over the vector and displaying the timestamp, open price, high price, low price, and closing price. The function returns the vector of Candlestick objects.

In conclusion, although I was only able to complete Task 1 of the assignment due to time management issues, I have made progress in setting up the necessary components for the trading system. Task 1 involved creating the Candlestick class, implementing its constructor, and defining the functions to handle candlestick statistics and display known products. While this submission may be incomplete, it represents the work accomplished within the given time frame. I apologize for not fulfilling all the tasks as originally intended, but I wanted to submit the progress I made.