Object Oriented Programming Coursework for Midterm: MerklerexBot

Introduction

In the course, we developed the Merklerex exchange simulation and a command-line user interface which allows a human to carry out trading activities such as bidding for and offering trades, market analysis and so on.

In this assignment you are tasked with creating a bot which can automatically carry out trading activities on the Merklerex platform. Essentially the bot takes the place of the human, interacting with the exchange to do the things that you were previously doing manually using the options on the menu.

Requirements

The purpose of the bot is to automate trading. The bot should start with a certain amount of currency and it should trade automatically with an aim to increasing the amount of currency it owns. The functionality of the bot should meet the following requirements:

- R1: Market analysis:
  - R1A: Retrieve the live order book from the Merklerex exchange simulation
  - R1B: Generate predictions of likely future market exchange rates using defined algorithms, for example, linear regression.
- R2: Bidding and buying functionality:
  - R2A: Decide when and how to generate bids using a defined algorithm which takes account of the current and likely future market price
  - R2B: Pass the bids to the exchange for matching
  - R2C: Receive the results of the exchange's matching engine (which decides which bids have been accepted) which might involve exchanging assets according to the bid and the matching, and the cost of the exchange generated by the simulation
  - R2D: Using the live order book from the exchange, decide if it should withdraw its bids at any point in time
- R3: Offering and selling
  - R3A: Generate offers using a defined algorithm which takes account of the current and likely future market price and pass the offers to the exchange for matching
  - R3B: Pass the bids to the exchange for matching

    - R3C: Receive the results of the exchange's matching engine (which decides which offers have been accepted) which might involve exchanging assets according to the offer and the matching, and the cost of the exchange generated by the simulation
    - R3D: Using the live order book from the exchange, decide if it should withdraw its offers at any point in time
- R4: Logging
    - R4A: Maintain a record of its assets and how they change over time
    - R4B: Maintain a record of the bids and offers it has made in a suitable file format
    - R4C: Maintain a record of successful bids and offers it has made, along with the context (e.g. exchange average offer and bid) in a suitable file format
- **Challenge requirement**: the order and bid processing code on the exchange is not well optimised – with a large dataset, it becomes quite slow. Can you optimise the exchange code so that it runs faster? To complete this challenge, in your report, present information about the performance before and after optimisation, then describe how you optimised the exchange code. Make it clear how you tested the speed and where in the exchange code you made your edits.

Code style and technique

Your code should be written according to the following style and technique guidelines:

- C1: Code is organised into header (.h or .hpp) files and implementation files (.cpp). Header files contain class interface definitions, cpp files contain implementations of class function memenbers.
- C2: Class interfaces in header files have comments for each public function describing purpose, inputs and outputs
- C3: Code is laid out clearly with consistent indenting
- C4: Code is organised into functions with clear inputs and outputs and a clear, limited purpose
- C5: Code is stateless wherever possible – functions make use of data passing in preference to global or class scope data.
- C6: Functions, classes and variables have meaningful names, with a consistent naming style
- C7: Functions do not change the state of class or global scope variables unless that is the explicit purpose of a function (e.g. a setter)

Documentation

You should write a report and submit your source code. The submission should contain the following items and information:

- D1: Source code in standard ZIP format
- D2: Report in PDF format
- D3: Requirements: for each sub requirements (R1A → R4C) state how this was achieved or if it was not achieved. Explain where it can be found in the code. Use focused, short code extracts if they make your explanation clearer.

- D4: Algorithms: where you created algorithms, e.g. for predicting future market, explain each algorithm and state where it can be found in the code. Use focused, short code extracts if they make your explanation clearer.
- Challenge requirement: Document the optimisation you made to the exchange code, if you did this, as described above.

Marking criteria

We will mark your work according to the set of criteria shown below, which consider the requirements, your programming technique and style and the documentation you have provided:

Category	Criterion	Not addressed	Attempted but did not meet requirements	Met requirements well but did not go beyond that	Met requirements well and went significantly beyond them.
Code style and technique	C1: Code is organised into header (.h or .hpp) files and implementation files (.cpp). Header files contain class interface definitions, cpp files contain implementations of class function memenbers.				
Code style and technique	C2: Class interfaces in header files have comments for each public function describing purpose, inputs and outputs				
Code style and technique	C3: Code is laid out clearly with consistent indenting				
Code style and	C4: Code is organised into				

technique	functions with clear inputs and outputs and a clear, limited purpose				
Code style and technique	C5: Code is stateless wherever possible – functions make use of data passing in preference to global or class scope data.				
Code style and technique	C6: Functions, classes and variables have meaningful names, with a consistent naming style				
Code style and technique	C7: Functions do not change the state of class or global scope variables unless that is the explicit purpose of a function (e.g. a setter)				
R1: Market analysis	R1A: can retrieve order book				
	R1B: can generate predictions of future prices				
R2: Bidding and buying	R2A: Decide when to generate bids				
	R2B: Register bids on the exchange				
	R2C: Receive results of				

	matching and update assets accordingly				
	R2D: Decide when to withdraw bids and withdraw them				
R3: Offering and selling	R3A: Decide when to generate offers				
	R3B: Register offers on the exchange				
	R3C: Receive results of matching and update assets accordingly				
	R3D: Decide when to withdraw offers and withdraw them				
R4: Logging	R4A: generate asset log file				
	R4B: generate offer and bid log file				
	R4C: generate purchase and sale log file along with context data				
Challenge requireme nt	Optimise the exchange code				
Document ation	D1: source code as zip				
	D2: report as PDF				
	D3: reporting on requirements				
	D4: description of				

	algorithms				
Challenge requirement	Describe performance before and after your optimisations. Describe your optimisations				