Backtesting Trading Strategies using OOP concepts in C++

Arundhati Sinha
Ashutosh
Shaunak Gupta

December 2022

Abstract

[1] Backtesting is a key component of effective trading system development. It is accomplished by reconstructing, with historical data, trades that would have occurred in the past using rules defined by a given strategy. The result offers statistics to gauge the effectiveness of the strategy.

The underlying theory is that any strategy that worked well in the past is likely to work well in the future, and conversely, any strategy that performed poorly in the past is likely to perform poorly in the future. This report cators to a scalable project where several trading strategies are backtested using historical stock prices using object oriented programming concepts using C++ language.

Contents

1 Objective

The main aim of the project is to understand C++ concepts of object oriented programming in finance by implementing a model of quantitative trading strategies using mean reversion and technical indicator strategies through the concept of backtesting. Backtesting is the process of poring over historical price data and testing your strategy over hundreds or thousands of past trading opportunities in order to collect statistics about your edge.

Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

The project has been divided into major sections for clarity of flow and understanding which includes:

1. [2]Collection and cleaning of data through relevant research and reliable resources.
2. Application and understanding of technical indicators and trading strategies.
3. Implementing the logic in C++ using OOP concepts.

2 Trading Strategy

[3]Quantitative trading involves using quantitative methods and algorithms to execute strategies. This has a broad set of uses; a typical example might be a trader using a mathematical model to take a position on what an asset "should" be worth before carrying out a trade. Models like these are designed to leverage information the market may be missing, such as what happens to a company or an industry when interest rates move in a particular direction. However, this is only one form of quantitative trading, and is most often performed by hedge funds or investment banks.

2.1 Mean Reversion

Mean reversion trading in equities tries to capitalize on extreme changes in the pricing of a particular security, assuming that it will revert to its previous state. This theory can be applied to both buying and selling, as it allows a trader to profit on unexpected upswings and to save on abnormal lows.

2.2 MACD

Moving average convergence/divergence (MACD, or MAC-D) is a trend-following momentum indicator that shows the relationship between two exponential moving averages (EMAs) of a security's price. The MACD line is calculated by subtracting the 26-period EMA from the 12-period EMA.
The result of that calculation is the MACD line. A nine-day EMA of the MACD line is called the signal line, which is then plotted on top of the MACD line, which can function as a trigger for buy or sell signals. Traders may buy the security when the MACD line crosses above the signal line and sell—or short—the security when the MACD line crosses below the signal line. MACD indicators can be interpreted in several ways, but the more common methods are crossovers, divergences, and rapid rises/falls.

3 Model

Quantitative trading strategy model is designed with the principles of OOP in C++.

There are some basic concepts that act as the building blocks of OOPs:

(a) Object
An Object can be defined as an entity that has a state and behavior, or in other words, anything that exists physically in the world is called an object. It can represent a dog, a person, a table, etc. An object means the combination of data and programs, which further represent an entity.

(b) Classes
Class can be defined as a blueprint of the object. It is basically a collection of objects which act as building blocks. A class contains data members (variables) and member functions. These member functions are used to manipulate the data members inside the class.

(c) Abstraction
Abstraction helps in the data hiding process. It helps in displaying the essential features without showing the details or the functionality to the user. It avoids unnecessary information or irrelevant details and shows only that specific part which the user wants to see.

(d) Encapsulation
The wrapping up of data and functions together in a single unit is known as encapsulation. It can be achieved by making the data members' scope private and the member function's scope public to access these data members. Encapsulation makes the data non-accessible to the outside world.

(e) Inheritance
Inheritance is the process in which two classes have an is-a relationship among each other and objects of one class acquire properties and features of the other class. The class which inherits the features is known as the child class, and the class whose features it inherited is called the parent class.

(f) Polymorphism
Polymorphism means many forms. It is the ability to take more than one form. It is a feature that provides a function or an operator with more than one definition. It can be implemented using function overloading, operator overload, function overriding, virtual function.

3.1 UML Diagram

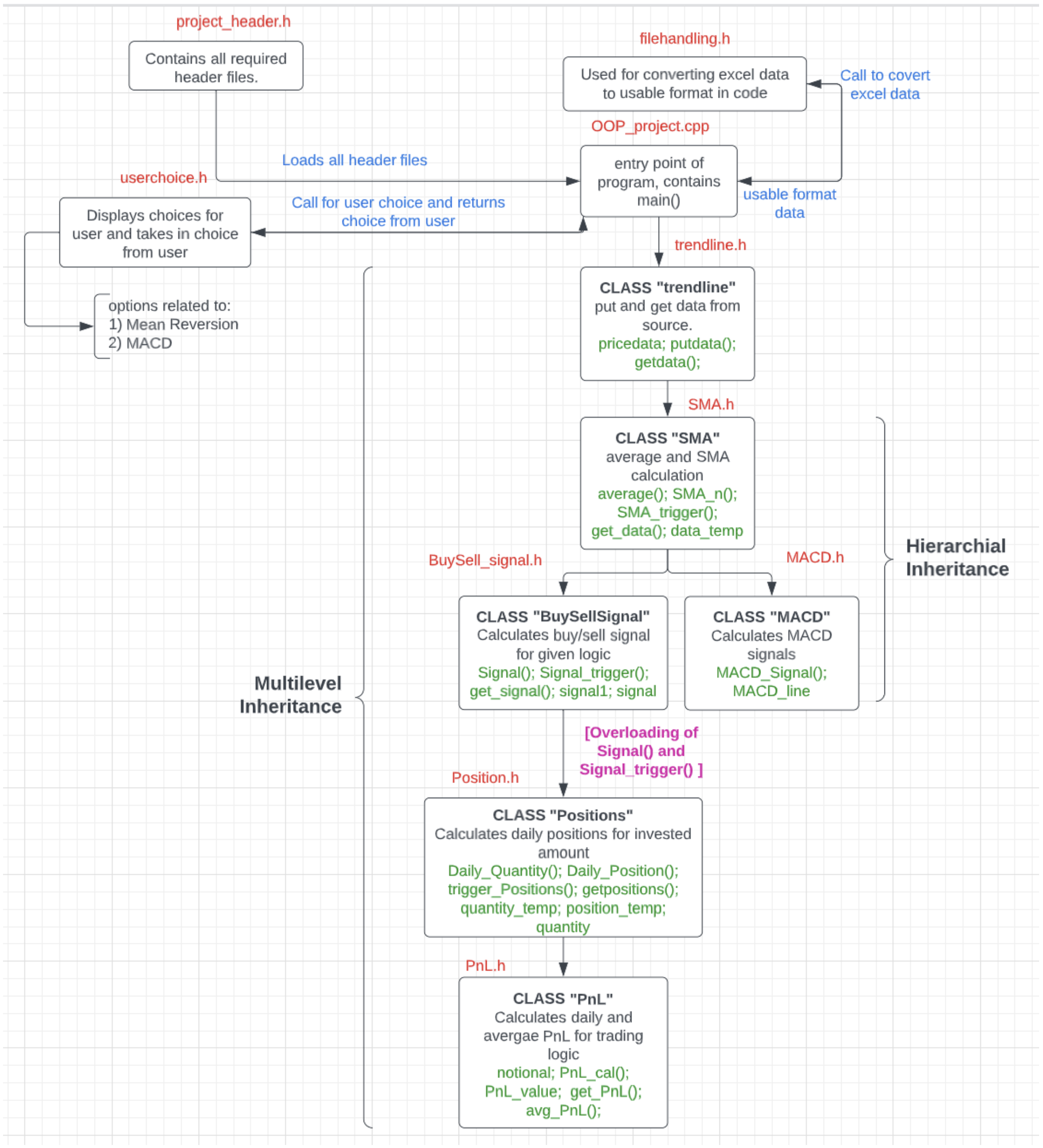

project_header.h
Contains all required header files.
filehandling.h
Used for converting excel data to usable format in code
Call to covert excel data
OOP_project.cpp
entry point of program, contains main()
Loads all header files
usable format data
userchoice.h
Displays choices for user and takes in choice from user
Call for user choice and returns choice from user
options related to: 1) Mean Reversion 2) MACD
trendline.h
CLASS "trendline" put and get data from source. pricedata; putdata(); getdata();
SMA.h
CLASS "SMA" average and SMA calculation average(); SMA_n(); SMA_trigger(); get_data(); data_temp
Hierarchial Inheritance
BuySell_signal.h
MACD.h
CLASS "BuySellSignal" Calculates buy/sell signal for given logic Signal(); Signal_trigger(); get_signal(); signal1; signal
CLASS "MACD" Calculates MACD signals MACD_Signal(); MACD_line
Multilevel Inheritance
[Overloading of Signal() and Signal_trigger()]
Position.h
CLASS "Positions" Calculates daily positions for invested amount Daily_Quantity(); Daily_Position(); trigger_Positions(); getpositions(); quantity_temp; position_temp; quantity
PnL.h
CLASS "PnL" Calculates daily and avergae PnL for trading logic notional; PnL_cal(); PnL_value; get_PnL(); avg_PnL();

Figure 1: UML diagram of project flow
green = member variables and functions
blue = flow logic
red = filename
pink = additional comments

3.2 Files Explanation

This section of the report will explain the division of files in the project along with their role in the overall code.

1. project_header.h

   This file is a dump of all required header files required for a zero error code.

2. userchoice.h

   This file deals with user choices on what they want to achieve through the code and sends the details to other files on the basis of user selection.

3. filehandling.h

   The file is made to read price data from an external excel file and covert that data into readable usable format using fstream functionalities in C++.

   The price data is stored in vector "price_temp" and the date data is stored in vector "date_temp".

4. trendline.h

   This file deals with management of raw data from excel files. It contains a class "trendline" with two member functions:

   (a) put_data( ) = to store price data into vector pricedata of type float. (public)
   (b) get_data( ) = to fetch price data from vector pricedata. (public)

   This file is also made with the scope to include matplotlib functionalities for user friendly graph feature in C++.

5. SMA.h

   The file deals with calculation of SMA values. SMA or Simple Moving Average is simply the average price over the specified period. The average is called "moving" because it is plotted on the chart bar by bar, forming a line that moves along the chart as the average value changes. The file contains a class "SMA" which is a derived class from base class trendline. SMA class contains the following member functions:

   (a) average( ) = to calculate average value of "n" days that is used in calculation of SMA signal. (private)
   (b) SMA_n( ) = a function to calculate SMA signal. (private)
   (c) data_temp = a local vector variable of type float to store SMA values. (public)
   (d) SMA_trigger( ) = used to trigger internal calculation functions of average( ) and SMA_n( ). (public)
   (e) get_data( ) = function to get data from data_temp variable.(public)

6. BuySell_signal.h

   This file generates buy and sell signal using the trading logic provided to the file. It uses function overloading to generate signals for different strategies based on input signature for functions. The file contains a class "BuySellSignal" which is a derived class from base class SMA. BuySellSignal class contains the following member functions:

   (a) Signal( ) = function used to generate buy/sell signals for given strategy. The function in overloaded for different strategies. (private)
   (b) Signal_trigger( ) = used to trigger private functions and initiate variables. (public)
   (c) getsignal( ) = used to get signal data from float vector signal. (public)
   (d) signal = float vector to store signal data. (public)

7. MACD.h

   This file generates signals for MACD technical indicator which is later used through BuySellSignal class to generate buy/sell signals. The file contains a class "MACD" which is a derived class from base class SMA. MACD class contains the following member functions:

   (a) MACD_line = float vector to store MACD line values. (private)
   (b) MACD_Signal( ) = function which contains logic to calculate MACD_line. (private)
   (c) getsignal( ) = function to fetch MACD line values from MACD_line vector. (public)

8. Position.h

   The file is used to compute day-to-day positions on the investment amount with regards to buy and sell quantities taken as per price and signal data. The file contains a class "Positions" which is a derived class from base class BuySellSignal. Positions class contains the following member functions:

   (a) Daily_Quantity( ) = Calculates quantity to be investment at each buy/sell signal and at that given price. (private)
   (b) Daily_Position = Calculates day-to-day positions in accordance with quantity calculated. (private)
   (c) quantity_temp = temporary float vector to store calculated quantity. (public)
   (d) position_temp = temporary float vector to store calculate position. (public)
   (e) trigger_Positions( ) = function to invoke private functions for calculations. (public)
   (f) getpositions( ) = function to fetch position data from position_temp vector. (public)

9. PnL.h

   The file is responsible for calculation of day-to-day and average Proft and Loss from the trading strategy. The file contains a class "PnL" which is a derived class from base class Postions. PnL class contains the following member functions:

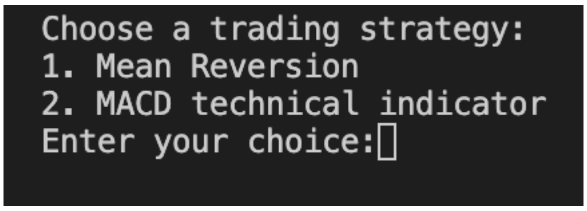
   (a) notional = int vector to store notional value. It is the total value of the position. (private)
   (b) PnL_cal( ) = function used to calculate daily PnL. (private)
   (c) PnL_value = int vector to store Profit and losses. (public)
   (d) trigger_PnL = function to invoke private functions for calculations. (public)
   (e) getPnL( ) = function to fetch Profit and loss data from vector PnL_value. (public)
   (f) avgPnL( ) = function to calculate average PnL from the trading cycle. (public)

10. Additional details:

   (a) Every .h file is protected with include guard. In the C and C++ programming languages, an #include guard, sometimes called a macro guard, header guard or file guard, is a particular construct used to avoid the problem of double inclusion when dealing with the include directive.

4 Output

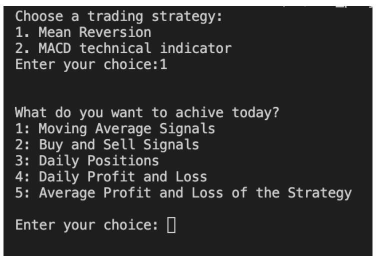
1. The user is first given a choice to choose the trading strategy.



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:
```

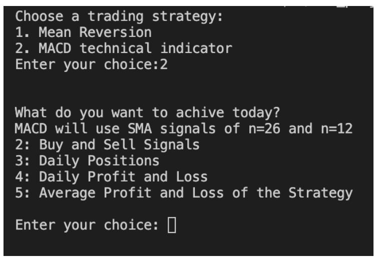
Figure 2: First Input from user

input ”1” = Mean Reversion Strategy
input ”2” = MACD Strategy
any other input = input error



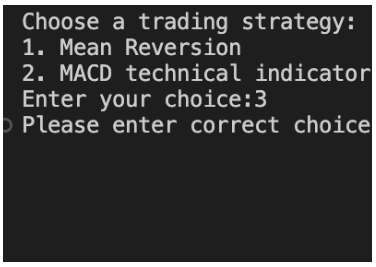
```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 
```

Figure 3: input ”1”



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:2


What do you want to achive today?
MACD will use SMA signals of n=26 and n=12
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 
```

Figure 4: input ”2”



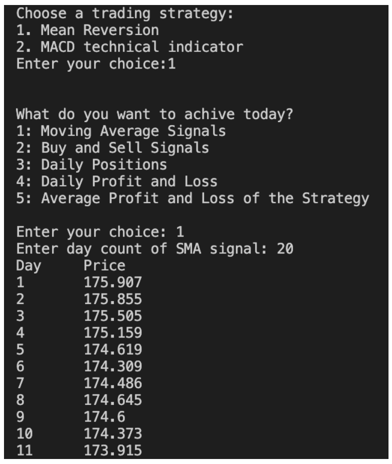
```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:3
Please enter correct choice
```

Figure 5: Invalid input

2. After selection of the trading strategy the user if given options to achieve data from different parts of the strategy as per his/her choice of task.

   (a) Mean Reversion

      i. Moving Average Signals



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 1
Enter day count of SMA signal: 20
Day     Price
1       175.907
2       175.855
3       175.505
4       175.159
5       174.619
6       174.309
7       174.486
8       174.645
9       174.6
10      174.373
11      173.915
```

Figure 6: Moving Average Signals

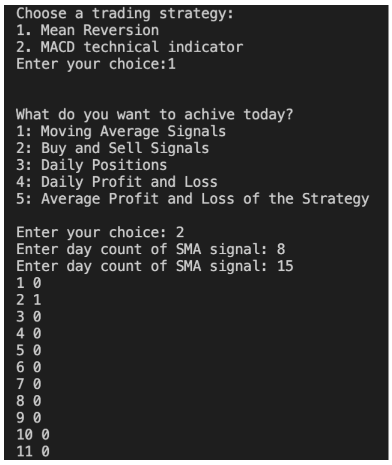
ii. Buy and Sell Signals

The signals have the following indication:

1 = Buy

-1 = Sell

0 = Do nothing

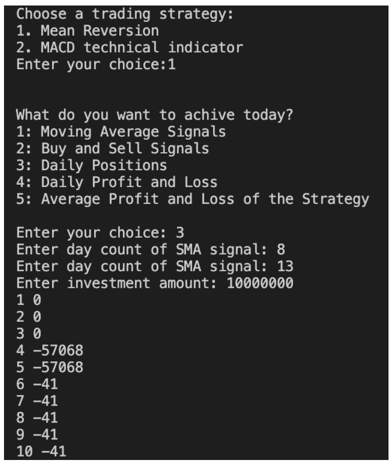


```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 2
Enter day count of SMA signal: 8
Enter day count of SMA signal: 15
1 0
2 1
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
```

Figure 7: Buy and Sell Signals

iii. Daily Positions

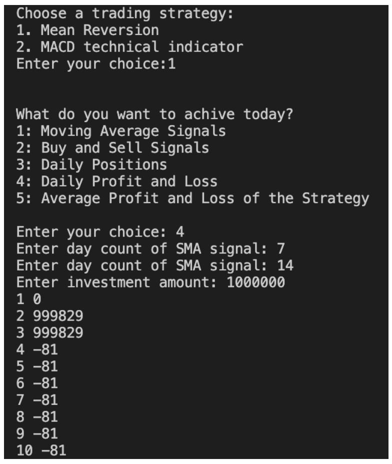
Gives out data for daily Position of the user with investment amount.



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 3
Enter day count of SMA signal: 8
Enter day count of SMA signal: 13
Enter investment amount: 10000000
1 0
2 0
3 0
4 -57068
5 -57068
6 -41
7 -41
8 -41
9 -41
10 -41
```

Figure 8: Daily Positions

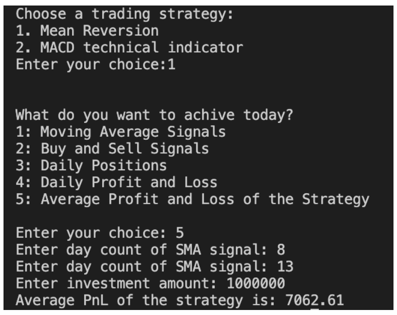
iv. Daily Profit and Loss



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 4
Enter day count of SMA signal: 7
Enter day count of SMA signal: 14
Enter investment amount: 1000000
1 0
2 999829
3 999829
4 -81
5 -81
6 -81
7 -81
8 -81
9 -81
10 -81
```

Figure 9: Daily Profit and Loss

v. Average Profit and Loss of the Strategy

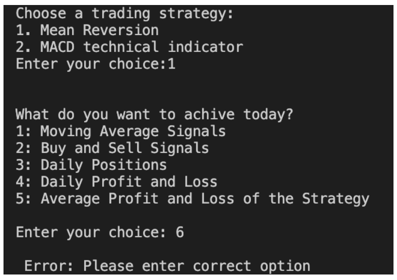


```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 5
Enter day count of SMA signal: 8
Enter day count of SMA signal: 13
Enter investment amount: 1000000
Average PnL of the strategy is: 7062.61
```

Figure 10: Average Profit

vi. Invalid Input

The code checks for invalid input.



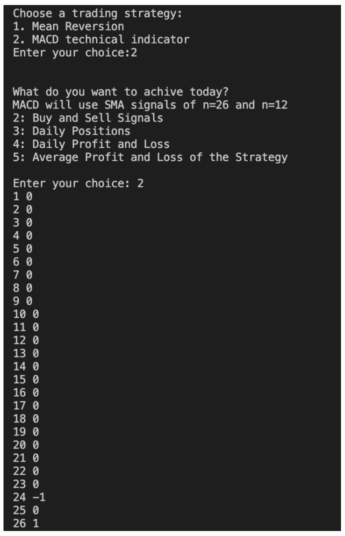
```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:1


What do you want to achive today?
1: Moving Average Signals
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 6

 Error: Please enter correct option
```

Figure 11: Invalid Input

(b) MACD [Moving Average Convergence/Divergence (MACD)]

i. Buy and Sell Signals

The signals have the following indication:
1 = Buy
-1 = Sell
0 = Do nothing

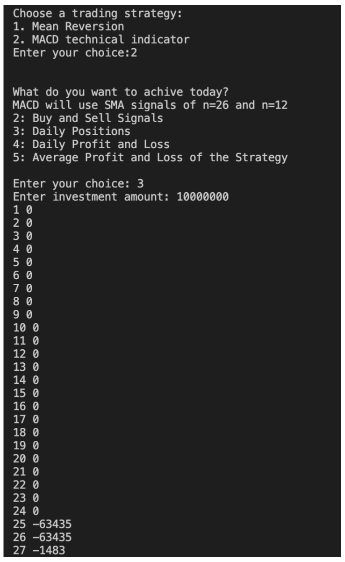


```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:2


What do you want to achive today?
MACD will use SMA signals of n=26 and n=12
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 2
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
21 0
22 0
23 0
24 -1
25 0
26 1
```

Figure 12: Buy and Sell Signals

ii. Daily Positions

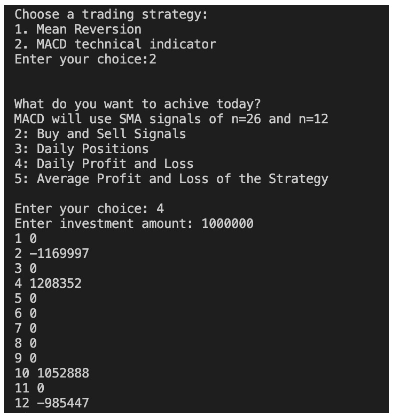
Gives out data for daily Position of the user with investment amount.



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:2


What do you want to achive today?
MACD will use SMA signals of n=26 and n=12
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 3
Enter investment amount: 10000000
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
21 0
22 0
23 0
24 0
25 -63435
26 -63435
27 -1483
```

Figure 13: Daily Positions

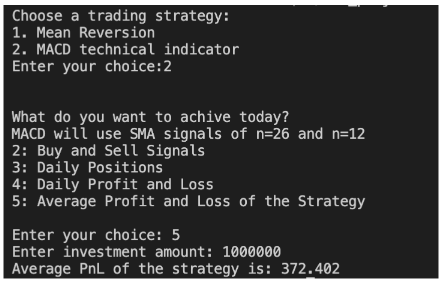
iii. Daily Profit and Loss



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:2


What do you want to achive today?
MACD will use SMA signals of n=26 and n=12
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 4
Enter investment amount: 1000000
1 0
2 -1169997
3 0
4 1208352
5 0
6 0
7 0
8 0
9 0
10 1052888
11 0
12 -985447
```

Figure 14: Daily Profit and Loss

iv. Average Profit and Loss of the Strategy

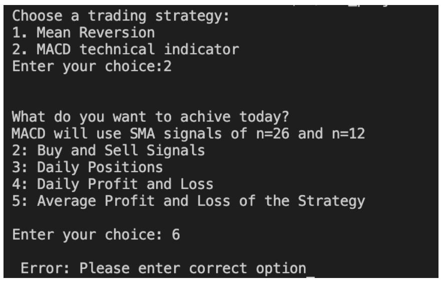


```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:2


What do you want to achive today?
MACD will use SMA signals of n=26 and n=12
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 5
Enter investment amount: 1000000
Average PnL of the strategy is: 372.402
```

Figure 15: Average Profit

v. Invalid Input

The code checks for invalid input.



```
Choose a trading strategy:
1. Mean Reversion
2. MACD technical indicator
Enter your choice:2


What do you want to achive today?
MACD will use SMA signals of n=26 and n=12
2: Buy and Sell Signals
3: Daily Positions
4: Daily Profit and Loss
5: Average Profit and Loss of the Strategy

Enter your choice: 6

 Error: Please enter correct option_
```

Figure 16: Invalid Input

5 Future Scope and Improvement

(a) Error handling can be done better.

(b) Scale of the project can be increased to more strategies.

(c) Optimization of code in terms of space and performance.

(d) Testing the trading logic on live and constant data.

6 Result

The project exposes us to varied understandings of OOP concepts and backtesting trading strategies.

The project cators to the following learnings:

1. Research about the stock and collect/clean the data to be evaluated.
2. Research about trading strategies and their performance factors.
3. Study and implementation of OOP concepts using C++.

A project on backtesting of quantitative trading strategies was succesfully implemented using object oriented concepts in C++.

References

[1] Investopedia. [Online]. Available: https://www.investopedia.com/

[2] Yahoo finance. [Online]. Available: https://finance.yahoo.com

[3] Imc quant trading. [Online]. Available: https://careers.imc.com/us/en/blogarticle/what-is-quantitative-trading?utm_source=google&utm_medium=paid&utm_campaign=us&utm_term=&utm_content=574826780522&gclid=Cj0KCQiA4aacBhCUARIsAI55maFtTRZIDMMLGaYcGA9VTCEGH7wh68vttrMZavTSE1isNHkfp1zlctYaAp8wEALw_wcB

7 Appendix

Code of the project has been submitted in files.
Code can also be accessed here on github branch:
`https://github.com/ashutoshrana171/OOP_Project/tree/OOP_Project_Final`