**Aim of the Project:**

In your language of choice, write a web service that takes in a stock symbol (e.g. AAPL, GOOG) and returns a single buy and sell date that yields the max profit over the last 180 days. You may use any third-party api to retrieve historical stock prices, but only consider the high price for the day.

**Solution:**

1. Design
   * End to end implementation includes a RESTful web service invoking a third-party API to retrieve historical prices for a stock.
   * An example query (Select any symbol and replace **GOOG** below) –
     + [http://localhost:8080/MaxProfit/maxprofit?symbol=**GOOG**](http://localhost:8080/MaxProfit/maxprofit?symbol=GOOG)
   * Sample output for the query above –
     + For Maximum Profit with GOOG in the last 180 days:

Buy Date = 2018-04-03 Sell Date = 2018-07-27 Maximum Profit = $252.90

1. Third party API
   * Third party API called Alpha Vantage API is used to fetch historical stock prices for any stock.
   * API Limitations:
     + Data set offered by API can be either compact (100 data points = 100 days) or full (daily prices for 20 years).
       - Since data required is for last 180 days, using date filter in full output size, data has been obtained for 124 valid days in the 180-day period.
     + API returns these 124 data points (daily high prices) as a JSON object with daily prices & other information about the stock.
       - As JSON object doesn’t retain the ordering of elements within the object, elements have been sorted by date using a TreeMap data structure causing the performance of the implementation to lower due to increase in both, time & space complexity.
       - Above overhead could’ve been avoided if the API returned JSON Array instead of JSON object.
     + Error Handling for data obtained from API
       - API has a limitation of returning the same response code – 200 for successful as well as unsuccessful data retrieval.
       - However, if a situation occurs in which a different response code is returned (in case of a failed/error data request), the response code should be handled successfully.
       - Currently, API responds with a generic error message in case of an invalid query. Following two error handling scenarios have been handled –
         * If stock symbol passed is null, empty or whitespace
         * If an invalid stock symbol is passed.
2. Algorithm
   * Assumptions
     + Since only high price in a day is considered for a stock, daily data has been used.
     + As only one value of a stock is available in a day, no intra-day trading is allowed. Only one transaction is allowed per day either buy or sell.
     + Only one unit (share) of a stock is considered for buying or selling.
     + A stock unit must be bought before it is sold. Since, only one transaction is allowed per day, stock unit can only be bought on the first day but won’t be sold.
     + Buying stock unit on the last day doesn’t make sense as there is no time left to sell it.
   * Logic:
     + High price for the stock unit on the first day is the current cost price to begin with.
     + High price of the stock unit on every following day is compared with the current cost price.
     + If it is lower than the current cost price, the current high price is the new current cost price and the date is noted as potential buy date.
     + If the current price is greater than the cost price, profit is calculated by subtracting cost price from sell price.
     + If profit is greater than current maximum profit, then new profit is the current maximum profit which is noted along with the buy date and sell date.
     + This is executed till all entries in the TreeMap are traversed.
     + Time Complexity: O(n)
     + Space Complexity: O(n)
     + Total time duration for end to end execution ~ 2secs (should get printed.
3. Testing
   * End to end testing
     + Tested scenario for receiving an erred JSON object for a stock.
     + Tested for an empty stock symbol.
     + Tested for an invalid stock symbol.
   * Algorithm testing
     + Tested for stock with prices having an upward trend.
     + Tested for stock with prices having a downward trend.
     + Tested for stock with prices having downward spiral.
     + Tested for stock with randomly increasing and decreasing prices.