CM2005 – Object Oriented Programming Midterm assignment

Code style (i.e., appropriate indentation and descriptive comments) and OOP concepts for modular & reusable code.

- I used several Classes, Constructors, Overloaded functions, loaded Objects, plots and data changes when objects are changed, exception handling, encapsulation (Class for prediction, Candlestick, WeatherBook, and weatherBookEntery)

Please note I will make this report as simple as possible to read – I will take out only necessary data so you can mark each section.
The Program is much more complex that each section as I did not use the starter code
- I built all parts of the program from scratch with insparation from the MerkelMain app we made during the lectures

TASK 1: Compute candlestick data

```
WeatherBook::WeatherBook(std::string filename) {
    hourlyLogs = CSVReader::readCSV(filename);
}

std::vector<Candlestick> WeatherBook::getCandlestickPerYear() {

    //associate 1 string with a list of strings - https://www.youtube.com/watch?
v=aEgG4pidcKU
    std::map<std::string, std::vector<double>> tempsPerYear;
    for (const auto& entry : hourlyLogs) {
        std::string year = entry.timestamp.substr(0, 4);
        tempsPerYear[year].push_back(entry.AT_temperature);
    }

    std::vector<Candlestick> candlestickBook;

    std::string year{};
    double open{ };
    for (auto pair : tempsPerYear) {
        double high{ };
        double low{ };
        double avg{ };

        year = pair.first;

        for (auto temps : pair.second) {
            avg += temps;
            if (temps > high) {
                high = temps;
            }
            if (temps < low) {
                low = temps;
            }
        }
        avg = avg / pair.second.size();
        candlestickBook.push_back(Candlestick(year + "-01-01", open, high, low, avg));
        open = avg;
    }

    return candlestickBook;
}

void WeatherMain::printCandlestickData() {

      // Align and fill https://www.udemy.com/course/beginning-c-plus-plus-
programming/learn/lecture/10154724#overview
```

```
        const int fieldWidth1{ 12 };
        const int fieldWidth2{ 10 };
        std::cout << std::setw(fieldWidth1) << std::left
              << "Date" << std::setw(fieldWidth2) << std::left <<
              "Open" << std::setw(fieldWidth2) << std::left <<
              "High" << std::setw(fieldWidth2) << std::left <<
              "Low" << std::setw(fieldWidth2) << std::left <<
            "Close" << std::setw(fieldWidth2) << std::left << std::endl;

        for (Candlestick candle : candlestickBook) {
              std::cout << std::setw(fieldWidth1) << std::left << candle.year <<
                    std::setw(fieldWidth2) << std::left << candle.open <<
                    std::setw(fieldWidth2) << std::left << candle.high <<
                    std::setw(fieldWidth2) << std::left << candle.low <<
                    std::setw(fieldWidth2) << std::left << candle.close << std::endl;
        }
}
```

TASK 1: Compute candlestick data DESCRIPTION

This task has 3 parts:
Part 1 – load the data is a structured way:
1. Load each line to memory from file
2. Tokenise the lines into parts cutting lines based on “,”
3. Finally push the tokenised data into an Object for each entry called WeatherEntry
- If you want to examine this code its in the CSVReader.cpp functions CSVReader::readCSV and CSVReader::tokenize functions
NOTE – I have created my own object (WeatherEntery), Member functions for collecting data as well as exception handling for collecting invalid data

Part 2:
The member function from the WeatherBook Class WeatherBook::getCandlestickData this function creates Candlestick objects and loads them into a vector
Formats all the data into a structured format (A map data structure of unique years and temperatures for those years
- Computes High and Low for each year – simple comparison if the temperature Item is smaller or bigger that the last biggest or smallest weather item make this the new high or low
- Computes average – Add all the temperatures of the year and divides the sum by the vector length.
- Pushes year, open, high, low and avg a vector or Candlestick objects called CandlestickBook
- After that a variable called open stores avg so it can be used to keep the open data member of the Candlestick for the next interaction pushed
NOTE – Use of data structures map, vector as well as objects used for functions and data

Part 3:
Finally visual the data in a way the user can that easily read.
- simple iteration the prints out the candlestickBook data.
NOTE – I used std::setw from the iomanip standard library – it is used to align data into chunks
Screen shot of how it looks:
- Well spaced
- Not the open of the first year is 0 as a default as it does not have a previous years data to build of of

```
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
============================================================
2
You chose: 2
Date        Open      High      Low       Close
1980-01-01  0         29.132    -14.507   6.04915
1981-01-01  6.04915   29.419    -16.219   7.19199
1982-01-01  7.19199   28.395    -15.084   7.56654
1983-01-01  7.56654   32.416    -18.098   8.05365
1984-01-01  8.05365   32.659    -13.338   6.83589
1985-01-01  6.83589   29.166    -22.705   6.57698
1986-01-01  6.57698   29.785    -20.601   7.12091
1987-01-01  7.12091   28.054    -25.301   6.5998
1988-01-01  6.5998    31.313    -13.019   7.6885
1989-01-01  7.6885    28.968    -10.969   8.06729
1990-01-01  8.06729   29.145    -11.347   8.06745
1991-01-01  8.06745   30.448    -15.978   6.92601
1992-01-01  6.92601   32.326    -13.679   8.0428
1993-01-01  8.0428    30.092    -17.096   7.39017
1994-01-01  7.39017   31.458    -12.935   8.69822
1995-01-01  8.69822   31.62     -14.468   7.33136
1996-01-01  7.33136   27.225    -20.283   5.97616
1997-01-01  5.97616   27.528    -12.095   7.32246
1998-01-01  7.32246   31.945    -15.665   7.84674
1999-01-01  7.84674   29.667    -16.172   7.76021
2000-01-01  7.76021   32.539    -16.425   8.7638
2001-01-01  8.7638    30.489    -17.771   7.67527
2002-01-01  7.67527   30.305    -16.596   8.4207
2003-01-01  8.4207    33.745    -17.103   8.11389
2004-01-01  8.11389   28.997    -15.015   7.29018
2005-01-01  7.29018   31.314    -19.122   6.97859
2006-01-01  6.97859   30.55     -21.521   7.61534
2007-01-01  7.61534   32.701    -10.583   8.54585
2008-01-01  8.54585   28.851    -11.306   8.35788
2009-01-01  8.35788   29.917    -15.502   7.89768
2010-01-01  7.89768   30.419    -18.188   6.76271
2011-01-01  6.76271   32.679    -13.038   8.45616
2012-01-01  8.45616   32.54     -17.832   8.20152
2013-01-01  8.20152   32.556    -15.969   7.67734
2014-01-01  7.67734   29.589    -9.64     9.06574
2015-01-01  9.06574   32.577    -11.174   9.05732
2016-01-01  9.05732   29.467    -12.179   8.39235
2017-01-01  8.39235   32.631    -14.708   8.33198
2018-01-01  8.33198   30.307    -17.596   9.21341
2019-01-01  9.21341   31.839    -10.777   9.19864
============================================================
1: Print help
```

TASK 2: Create a text-based plot of the candlestick data CODE*

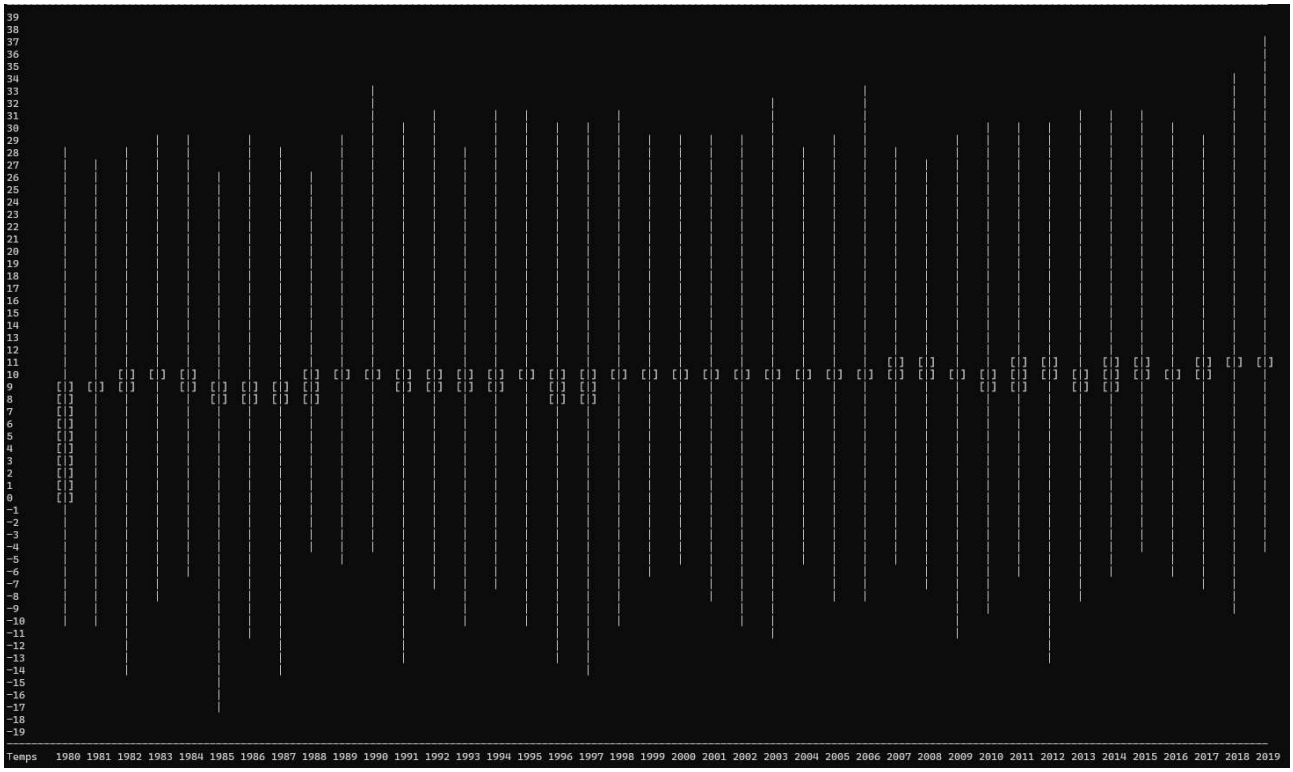
```
void WeatherMain::plotGraph() {

	double plotHeigh{};
	double plotLow{};
	for (const auto entry : candlestickBook) {
		if (entry.high > plotHeigh) {
			plotHeigh = entry.high;
		}
		if (entry.low < plotLow) {
			plotLow = entry.low;
		}
	}
	int buffer{ 2 };
	plotHeigh = static_cast<int>(plotHeigh) + buffer;
	plotLow = static_cast<int>(plotLow) - buffer ;
	std::string candlePlotText = "[|]";
	std::string stickPlotText = " | ";
	drawBoarder();

	std::cout << std::endl;
	std::cout << "Candle from low to high is represented by: " << candlePlotText << std::endl;
	std::cout << "Sick from low to high is represented by: " << stickPlotText << std::endl;
	drawBoarder();
	std::cout << std::endl;
	for (int j = plotHeigh; j >= plotLow; j-=1) {
```

```
                std::cout << std::setw(8) << std::left << j;
                for (int i = 0; i < candlestickBook.size(); ++i) {
                        if (j >= 0) {
                                if (candlestickBook[i].high >= j) {
                                        if ((std::round(candlestickBook[i].open) <= j &&
std::round(candlestickBook[i].close) >= j) || (std::round(candlestickBook[i].open) >= j &&
std::round(candlestickBook[i].close) <= j)) {
                                                std::cout << std::setw(5) << std::left << candlePlotText;
                                        }
                                        else {
                                                std::cout << std::setw(5) << std::left << stickPlotText;
                                        }

                                }
                                else {
                                        std::cout << std::setw(5) << std::left << "";
                                }
                        }
                        else {
                                if (candlestickBook[i].low <= j) {
                                        if ((std::round(candlestickBook[i].open) >= j &&
std::round(candlestickBook[i].close) <= j) || (std::round(candlestickBook[i].open) <= j &&
std::round(candlestickBook[i].close) >= j)) {
                                                std::cout << std::setw(5) << std::left << candlePlotText;
                                        }
                                        else {
                                                std::cout << std::setw(5) << std::left << stickPlotText;
                                        }

                                }
                                else {
                                        std::cout << std::setw(5) << std::left << "";
                                }
                        }
                }
                std::cout << std::endl;
        }
        drawBoarder();
        std::cout << std::endl;
        std::cout << std::setw(8) << std::left << "Temps";
        for (int i = 0; i < candlestickBook.size(); ++i) {
                        std::cout << std::setw(5) << std::left << candlestickBook[i].year.substr(0, 4);
        }
        std::cout << std::endl;
        drawBoarder();
        std::cout << std::endl;

}
```

TASK 2: Create a text-based plot of the candlestick data DESCRIPTION

1. Computes the maximum and minimum across all years and adds a buffer so it can create so it can build the plot maximums and minimums
2. Interacts through the code through the code for each year checks if not higher print blank if between open and close print [|] (the candle) and if between high and low prints | the stick

Note – the plot is dynamic and will work on any data and is well spaced so it is easily read:



TASK 3: Filtering option and plot a text graph CODE

Filter 1 (filter by year)

```
std::vector<Candlestick> WeatherBook::getCandlestickPerYear(std::string startYear, std::string
endYear) {

    std::map<std::string, std::vector<double>> tempsPerYear;
    for (const auto& entry : hourlyLogs) {
        std::string year = entry.timestamp.substr(0, 4);
        tempsPerYear[year].push_back(entry.AT_temperature);
    }

    std::vector<Candlestick> candlestickBook;

    std::string year{};
    double open{ };
    for (auto pair : tempsPerYear) {
        double high{ };
        double low{ };
        double avg{ };
        // skip loading these dates
        if (std::stoi(startYear) > std::stoi(pair.first) || std::stoi(endYear) <
std::stoi(pair.first)) {
            continue;
        }

        year = pair.first;

        for (auto temps : pair.second) {
            avg += temps;
            if (temps > high) {
                high = temps;
            }
            if (temps < low) {
                low = temps;
            }
        }
        //std::cout << year << " " << high << " " << low << std::endl;
        avg = avg / pair.second.size();
        candlestickBook.push_back(Candlestick(year + "-01-01", open, high, low, avg));
        open = avg;
    }
```

```
    for (Candlestick candle : candlestickBook) {
        std::cout << candle.year << " " << candle.open << " " << candle.high << " " << candle.low << "
" << candle.close <<  std::endl;
    }

    return candlestickBook;
}
```

Filter 2: (filter by country)

```
std::vector<Candlestick> WeatherBook::getCandlestickPerYear(std::string coutry) {
    //assosiate 1 string with a list of strings - https://www.youtube.com/watch?v=aEgG4pidcKU
    std::map<std::string, std::vector<double>> tempsPerYear;

    std::string coutryOption;
    if (coutry == "1") {
        for (const auto& entry : hourlyLogs) {
            std::string year = entry.timestamp.substr(0, 4);
            tempsPerYear[year].push_back(entry.AT_temperature);
        }
    }
    else if (coutry == "2") {
        for (const auto& entry : hourlyLogs) {
            std::string year = entry.timestamp.substr(0, 4);
            tempsPerYear[year].push_back(entry.BE_temperature);
        }
    }
    else if (coutry == "3") {
        for (const auto& entry : hourlyLogs) {
            std::string year = entry.timestamp.substr(0, 4);
            tempsPerYear[year].push_back(entry.BG_temperature);
        }
    }
    //default to AT_temperature
    else {
        for (const auto& entry : hourlyLogs) {
            std::string year = entry.timestamp.substr(0, 4);
            tempsPerYear[year].push_back(entry.AT_temperature);
        }
    }

    std::vector<Candlestick> candlestickBook;

    std::string year{};
    double open{ };
    for (auto pair : tempsPerYear) {
        double high{ };
        double low{ };
        double avg{ };

        year = pair.first;

        for (auto temps : pair.second) {
            avg += temps;
            if (temps > high) {
                high = temps;
            }
            if (temps < low) {
                low = temps;
            }
        }
        //std::cout << year << " " << high << " " << low << std::endl;
        avg = avg / pair.second.size();
        candlestickBook.push_back(Candlestick(year + "-01-01", open, high, low, avg));
        open = avg;
    }
    return candlestickBook;
}
```

TASK 3: Filtering option and plot a text graph DESCRIPTION

Filter 1: overloaded getCandileStickByYear member function that does not push data that is outside the years outside of the scope into the candlestickBook

Filter 2: Overloaded getCandileStickByYear function that changes the country that is loaded into the candlestickBook

Filter 1 – plot and data – by year

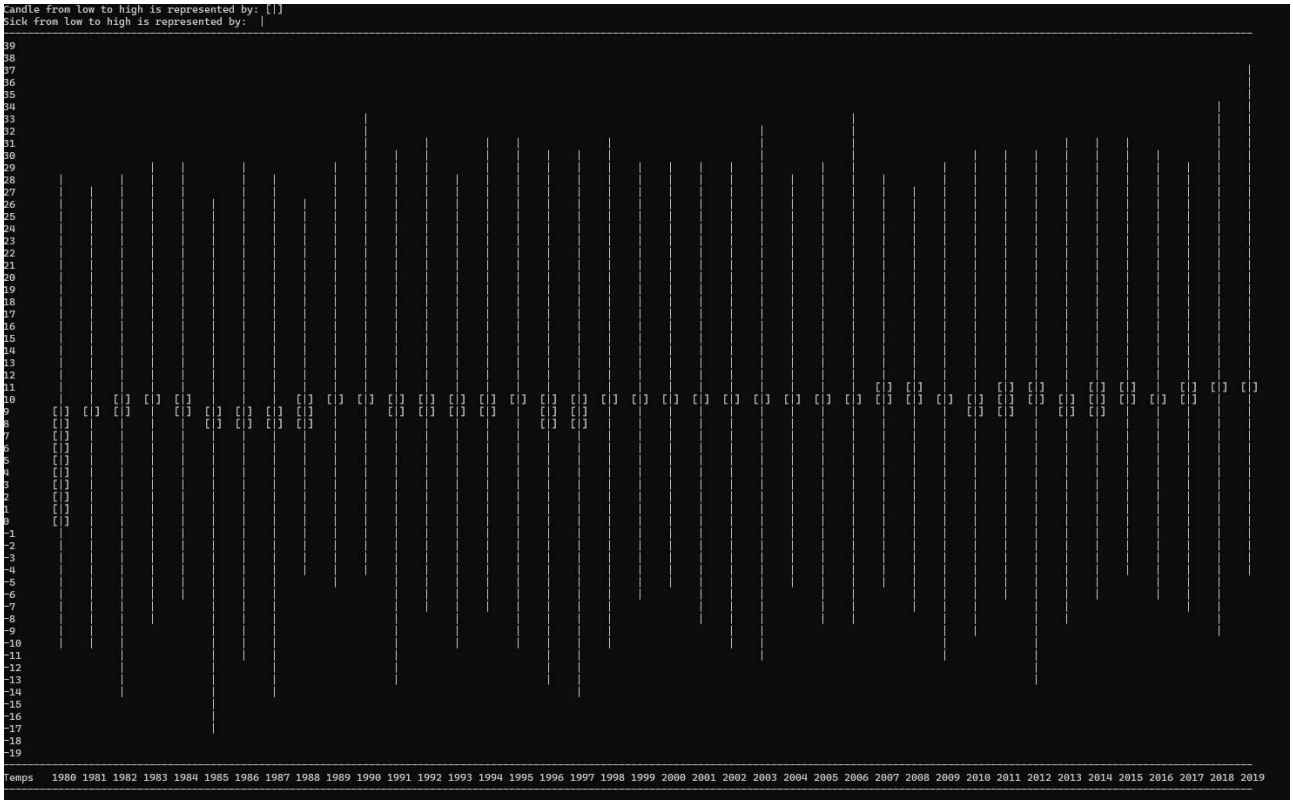
```
================================================================
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
================================================================
4
You chose: 4
Enter start year between 1980 and 2019
2000
Enter end year between 1980 and 2019
2010
2000-01-01 0 32.539 -16.425 8.7638
2001-01-01 8.7638 30.489 -17.771 7.67527
2002-01-01 7.67527 30.305 -16.596 8.4207
2003-01-01 8.4207 33.745 -17.103 8.11389
2004-01-01 8.11389 28.997 -15.015 7.29018
2005-01-01 7.29018 31.314 -19.122 6.97859
2006-01-01 6.97859 30.55 -21.521 7.61534
2007-01-01 7.61534 32.701 -10.583 8.54585
2008-01-01 8.54585 28.851 -11.306 8.35788
2009-01-01 8.35788 29.917 -15.502 7.89768
2010-01-01 7.89768 30.419 -18.188 6.76271
================================================================
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
================================================================
S
```

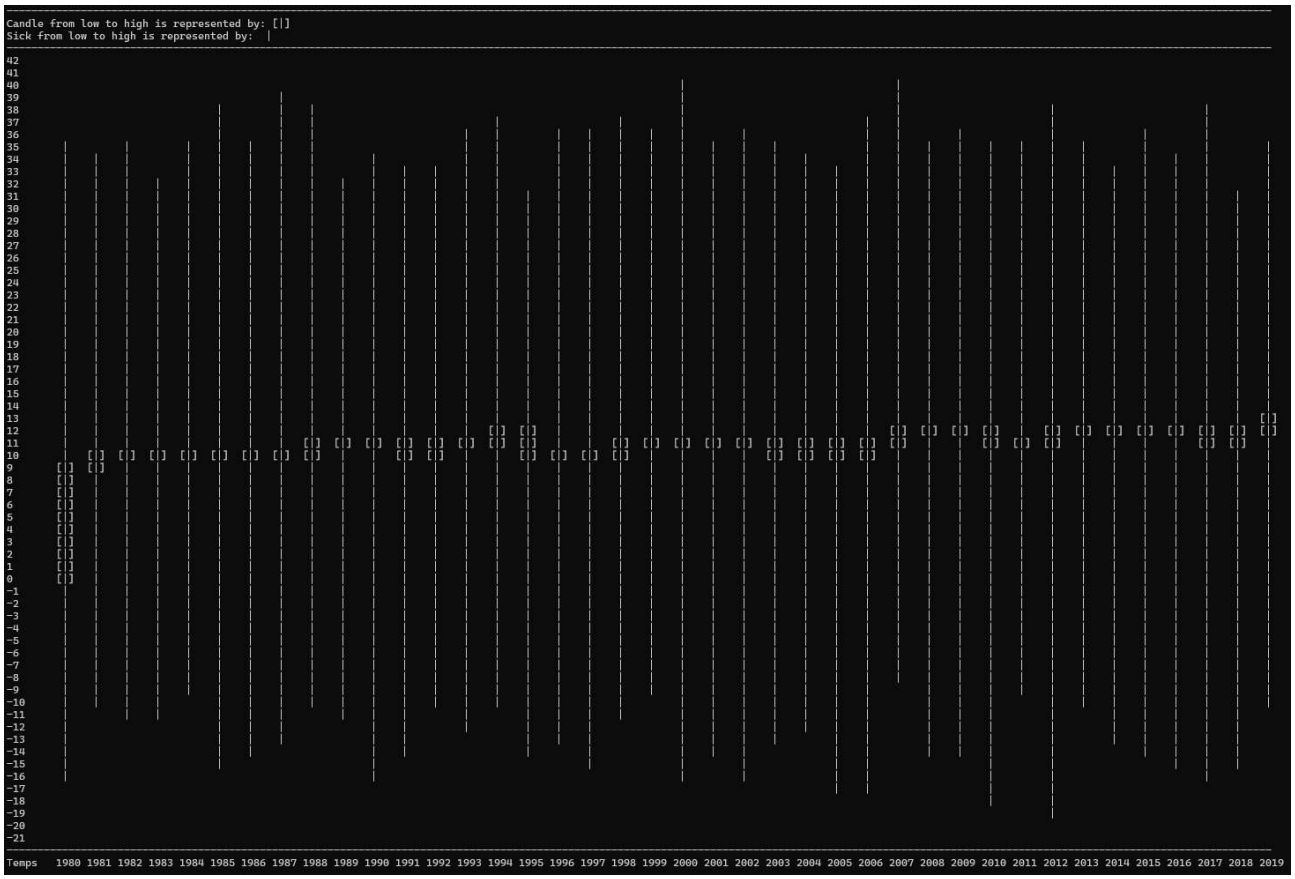
```
Candle from low to high is represented by: [|]
Sick from low to high is represented by:  |
------------------------------------------------------------
35
34
33                    |
32          |         |                   |
31          |         |         |         |
30          |   |     |         |   |     |               |
29          |   |     |         |   |     |         |     |
28          |   |     |   |     |   |     |   |     |     |
27          |   |     |   |     |   |     |   |     |     |
26          |   |     |   |     |   |     |   |     |     |
25          |   |     |   |     |   |     |   |     |     |
24          |   |     |   |     |   |     |   |     |     |
23          |   |     |   |     |   |     |   |     |     |
22          |   |     |   |     |   |     |   |     |     |
21          |   |     |   |     |   |     |   |     |     |
20          |   |     |   |     |   |     |   |     |     |
19          |   |     |   |     |   |     |   |     |     |
18          |   |     |   |     |   |     |   |     |     |
17          |   |     |   |     |   |     |   |     |     |
16          |   |     |   |     |   |     |   |     |     |
15          |   |     |   |     |   |     |   |     |     |
14          |   |     |   |     |   |     |   |     |     |
13          |   |     |   |     |   |     |   |     |     |
12          |   |     |   |     |   |     |   |     |     |
11          |   |     |   |     |   |     |   |     |     |
10          |   |     |   |     |   |     |   |     |     |
9       [|] [|]         |         |         |[|] [|]       |
8       [|] [|] [|] [|] [|] [|]     [|] [|] [|] [|] [|] [|]
7       [|]                 [|] [|] [|]                   [|]
6       [|]                 |                             |
5       [|]                 |                             |
4       [|]                 |                             |
3       [|]                 |                             |
2       [|]                 |                             |
1       [|]                 |                             |
0       [|]                 |                             |
-1        |                 |                             |
-2        |                 |                             |
-3        |                 |                             |
-4        |                 |                             |
-5        |                 |                             |
-6        |                 |                             |
-7        |                 |                             |
-8        |                 |                             |
-9        |                 |                             |
-10       |                 |                             |
-11       |                 |                             |
-12       |                 |                             |
-13       |                 |                             |
-14       |                 |                             |
-15       |                 |                             |
-16       |                 |                             |
-17       |                 |                             |
-18                         |                             |
-19                         |                             |
-20                         |                             |
-21                         |                             |
-22
-23
------------------------------------------------------------
Temps   2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010
------------------------------------------------------------
================================================================
```

Filter 1 – plot and data – by country (Austria and Belgium, others can be easily added)

```
Select coutry:
1: Austria
2: Belgium
3: Bulgaria
1
================================================================
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
================================================================
2
You chose: 2
Date        Open      High      Low       Close
1980-01-01  0         29.132    -14.507   6.04915
1981-01-01  6.04915   29.419    -16.219   7.19199
1982-01-01  7.19199   28.395    -15.084   7.56654
1983-01-01  7.56654   32.416    -18.098   8.05365
1984-01-01  8.05365   32.659    -13.338   6.83589
1985-01-01  6.83589   29.166    -22.705   6.57698
1986-01-01  6.57698   29.785    -20.601   7.12091
1987-01-01  7.12091   28.054    -25.301   6.5998
1988-01-01  6.5998    31.313    -13.019   7.6885
1989-01-01  7.6885    28.968    -10.969   8.06729
1990-01-01  8.06729   29.145    -11.347   8.06745
1991-01-01  8.06745   30.448    -15.978   6.92601
1992-01-01  6.92601   32.326    -13.679   8.0428
1993-01-01  8.0428    30.092    -17.096   7.39017
1994-01-01  7.39017   31.458    -12.935   8.69822
1995-01-01  8.69822   31.62     -14.468   7.33136
1996-01-01  7.33136   27.225    -20.283   5.97616
1997-01-01  5.97616   27.528    -12.095   7.32246
1998-01-01  7.32246   31.945    -15.665   7.84674
1999-01-01  7.84674   29.667    -16.172   7.76021
2000-01-01  7.76021   32.539    -16.425   8.7638
2001-01-01  8.7638    30.489    -17.771   7.67527
2002-01-01  7.67527   30.305    -16.596   8.4207
2003-01-01  8.4207    33.745    -17.103   8.11389
2004-01-01  8.11389   28.997    -15.015   7.29018
2005-01-01  7.29018   31.314    -19.122   6.97859
2006-01-01  6.97859   30.55     -21.521   7.61534
2007-01-01  7.61534   32.701    -10.583   8.54585
2008-01-01  8.54585   28.851    -11.306   8.35788
2009-01-01  8.35788   29.917    -15.502   7.89768
2010-01-01  7.89768   30.419    -18.188   6.76271
2011-01-01  6.76271   32.679    -13.038   8.45616
2012-01-01  8.45616   32.54     -17.832   8.20152
2013-01-01  8.20152   32.556    -15.969   7.67734
2014-01-01  7.67734   29.589    -9.64     9.06574
2015-01-01  9.06574   32.577    -11.174   9.05732
2016-01-01  9.05732   29.467    -12.179   8.39235
2017-01-01  8.39235   32.631    -14.708   8.33198
2018-01-01  8.33198   30.307    -17.596   9.21341
2019-01-01  9.21341   31.839    -10.777   9.19864
================================================================
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
================================================================
```

```
You chose: 5
More coutries can be added on request...
Select coutry:
1: Austria
2: Belgium
3: Bulgaria
2
================================================================
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
================================================================
2
You chose: 2
Date        Open      High      Low       Close
1980-01-01  0         28.304    -10.409   8.7913
1981-01-01  8.7913    27.849    -10.566   9.00897
1982-01-01  9.00897   28.711    -14.745   9.54383
1983-01-01  9.54383   29.359    -8.881    9.51549
1984-01-01  9.51549   29.397    -6.589    9.03268
1985-01-01  9.03268   26.648    -17.119   8.01571
1986-01-01  8.01571   29.064    -11.851   8.51663
1987-01-01  8.51663   28.007    -14.714   8.38263
1988-01-01  8.38263   26.403    -4.283    9.86974
1989-01-01  9.86974   29.935    -5.2      10.2686
1990-01-01  10.2686   33.529    -4.877    10.4372
1991-01-01  10.4372   30.671    -13.14    9.32394
1992-01-01  9.32394   31.848    -7.738    9.91921
1993-01-01  9.91921   28.287    -10.666   9.41135
1994-01-01  9.41135   31.32     -7.79     10.3027
1995-01-01  10.3027   31.294    -10.177   10.0509
1996-01-01  10.0509   30.235    -13.392   8.1897
1997-01-01  8.1897    30.32     -14.642   9.85406
1998-01-01  9.85406   31.861    -10.407   9.78459
1999-01-01  9.78459   29.033    -6.763    10.366
2000-01-01  10.366    29.174    -5.803    10.3652
2001-01-01  10.3652   29.656    -8.059    9.71412
2002-01-01  9.71412   29.847    -10.034   10.2056
2003-01-01  10.2056   32.779    -11.011   9.95233
2004-01-01  9.95233   28.235    -5.435    9.76304
2005-01-01  9.76304   29.691    -8.772    10.1719
2006-01-01  10.1719   33.196    -8.171    10.4168
2007-01-01  10.4168   28.333    -5.467    10.585
2008-01-01  10.585    27.733    -7.071    9.86716
2009-01-01  9.86716   29.15     -11.438   9.85279
2010-01-01  9.85279   30.645    -9.722    8.58514
2011-01-01  8.58514   30.263    -6.271    10.5366
2012-01-01  10.5366   30.057    -13.905   9.73195
2013-01-01  9.73195   31.125    -8.232    9.19143
2014-01-01  9.19143   31.059    -6.86     10.9556
2015-01-01  10.9556   31.027    -4.552    10.221
2016-01-01  10.221    30.251    -6.057    10.0232
2017-01-01  10.0232   29.965    -7.659    10.5302
2018-01-01  10.5302   34.277    -9.682    11.0965
2019-01-01  11.0965   37.918    -4.316    10.9766
================================================================
1: Print help
2: Print candlestick data
3: Plot candlestick text graph
4: Filter data per year
5: Change country
================================================================
```


Candle from low to high is represented by: [|]
Sick from low to high is represented by: |
Temps 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019


Candle from low to high is represented by: [|]
Sick from low to high is represented by: |
Temps 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019

TASK 4: Predicting data and plotting with a chosen model with CODE

```
#include "Prediction.h"


std::vector<Candlestick> Prediction::predictAverageMove(std::vector<Candlestick>
candleStickBook) {

	double closePrediction{};
	double lowPrediction{};
	double highPrediction{};
	double openPrediction{ candleStickBook[candleStickBook.size() - 1].close };
	int yearPrediction{ std::stoi(candleStickBook[candleStickBook.size() - 1].year)
+ 1 };
	std::string newYearPrediction = std::to_string(yearPrediction);
	newYearPrediction = newYearPrediction + "-01-01";
	for (Candlestick c : candleStickBook) {
		highPrediction += c.high;
		lowPrediction += c.low;
		closePrediction += c.close;
	}
	highPrediction = highPrediction / candleStickBook.size();
	lowPrediction = highPrediction / candleStickBook.size();
	closePrediction = closePrediction / candleStickBook.size();

	candleStickBook.push_back(Candlestick(newYearPrediction, openPrediction,
highPrediction, lowPrediction, closePrediction));

	return candleStickBook;
}
```

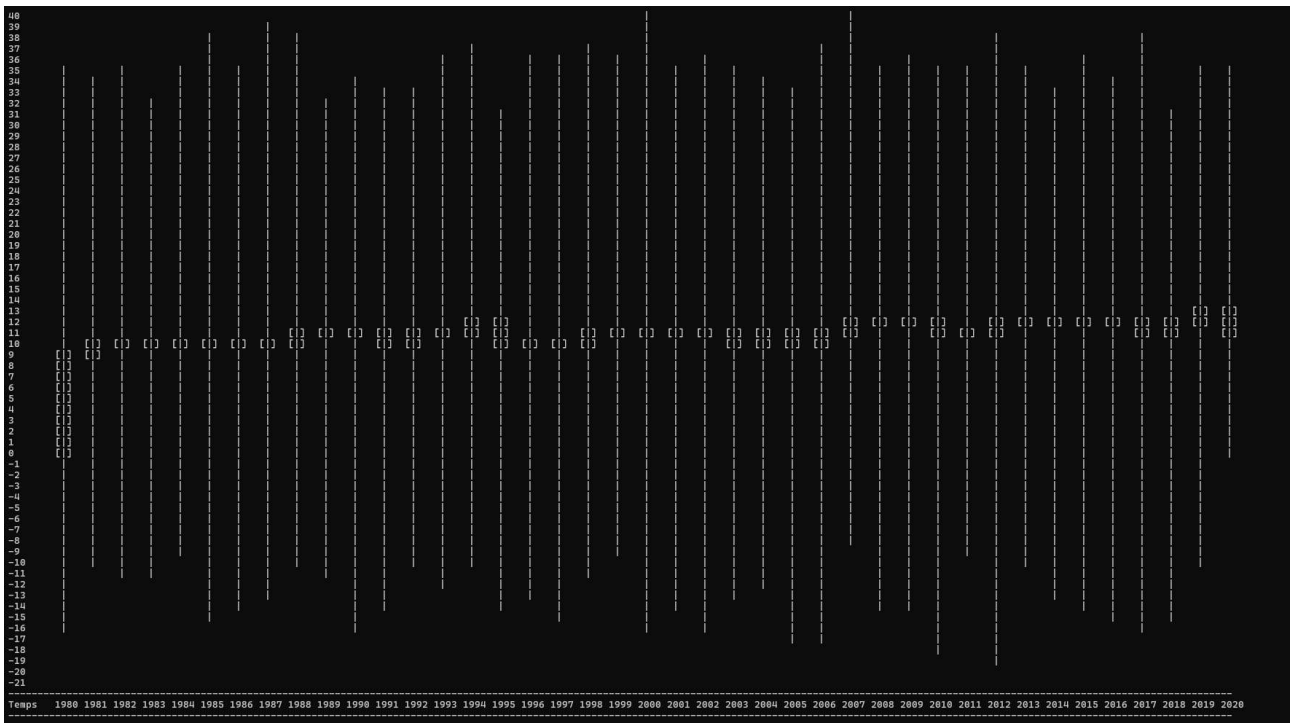
TASK 4: Predicting data and plotting with a chosen model justification and DESCRIPTION

https://www.investopedia.com/terms/m/movingaverage.asp

In simple average move is taking the average of the previous data using it as your next prediction

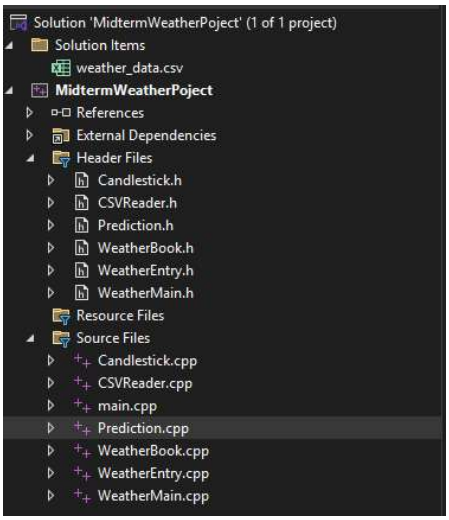
- I will calculate the average move for high, low and close the use it as my next prediction
- I decided to create an entire new class for prediction that takes in a vector of Candlesticks and returns then next years data
- I made use of a static member function

```
6: Predict data for next year for currenct dataset
============================================================
6
You chose: 6
Date        Open      High      Low       Close
1980-01-01  0         35.653    -16.576   9.43258
1981-01-01  9.43258   34.457    -10.616   10.226
1982-01-01  10.226    35.893    -11.141   10.1105
1983-01-01  10.1105   32.307    -11.482   10.1326
1984-01-01  10.1326   35.29     -9.433    9.9527
1985-01-01  9.9527    38.779    -15.775   10.0336
1986-01-01  10.0336   35.635    -14.435   10.4003
1987-01-01  10.4003   39.133    -13.917   10.0253
1988-01-01  10.0253   38.254    -10.948   10.5792
1989-01-01  10.5792   32.172    -11.087   10.8375
1990-01-01  10.8375   34.442    -16.967   11.4023
1991-01-01  11.4023   33.508    -14.452   9.90147
1992-01-01  9.90147   33.41     -10.967   10.55
1993-01-01  10.55     36.987    -12.869   10.558
1994-01-01  10.558    37.927    -10.968   11.9011
1995-01-01  11.9011   31.553    -14.952   10.1749
1996-01-01  10.1749   36.678    -13.638   9.95227
1997-01-01  9.95227   36.34     -15.128   9.75064
1998-01-01  9.75064   37.038    -11.423   10.7862
1999-01-01  10.7862   36.96     -9.678    11.4847
2000-01-01  11.4847   40.854    -16.486   11.433
2001-01-01  11.433    35.428    -14.537   11.1661
2002-01-01  11.1661   36.671    -16.604   11.0027
2003-01-01  11.0027   35.631    -13.61    10.3626
2004-01-01  10.3626   34.847    -12.051   10.8742
2005-01-01  10.8742   33.429    -17.843   10.232
2006-01-01  10.232    37.696    -17.178   10.822
2007-01-01  10.822    40.481    -8.074    12.0844
2008-01-01  12.0844   35.763    -14.106   11.7403
2009-01-01  11.7403   36.757    -14.146   11.5157
2010-01-01  11.5157   35.725    -18.714   11.3867
2011-01-01  11.3867   35.246    -9.042    10.7894
2012-01-01  10.7894   38.108    -19.326   11.7134
2013-01-01  11.7134   35.374    -10.778   11.9206
2014-01-01  11.9206   33.569    -13.029   11.6783
2015-01-01  11.6783   36.789    -14.714   11.6083
2016-01-01  11.6083   34.888    -15.58    11.6837
2017-01-01  11.6837   38.765    -16.119   11.4337
2018-01-01  11.4337   31.992    -15.902   11.872
2019-01-01  11.872    35.581    -10.885   12.7167
2020-01-01  12.7167   35.9002   0.897506  10.9057
```



Originality and the challenge of implementation

For originality I wrote all the code my self other than some copied code from MerkelMain which is clearly marked in the source code and did not use the starter code.
Even though I used MerkelMain for inspiration I still created the code and built the entire application from scratch
Used several classes and object where created with the pupose of this course to understand OOP



Loading and formatting the data was the hardest part of this project for me (integrating CSVReader into the application)
Spacing – using std::setw to nicely space object witch I learnt in a Udemy course

Not the separation of tasks low coupling and high cohesion

- Loading data – CSVReader Class – works as its own module
- Changing data – WeakerBook Class – helpful manipulating data in one class (filters and prediction)
- Displaying data – WeatherMain Class – used for user interaction

Clearly label all sections of the code that you personally wrote without assistance

I commented above all my code what was mine and what was copied from MerkelMain
Example:19

```
    return candlestickBook;
}
// Filter by coutry
// All my code - not he use of map and overloaded functions
std::vector<Candlestick> WeatherBook::getCandlestickPerYear(std::string coutry) {
    //assosiate 1 string with a list of strings - https://www.youtube.com/watch?v=aEgG4pidcKU
    std::map<std::string, std::vector<double>> tempsPerYear;

    std::string coutryOption;
    if (coutry == "1") {
        for (const auto& entry : hourlyLogs) {
            std::string year = entry.timestamp.substr(0, 4);
```