```cpp
#include <iostream>
#include <vector>
#include <string>
#include <map>
#include <limits>
#include <cmath>
#include <algorithm>
#include <iomanip>

#include "Utils.h"
#include "Candlestick.h"

/**
 * The main entry point of the program.
 *
 * @return Returns 0 if the program executes successfully, or 1 if an
error occurs.
 *
 * This program performs various tasks:
 * 1. Reads and validates a CSV file for the weather dataset.
 * 2. Computes candlestick data for a country of my choice.
 * 3. Plots candlestick data (grouped by decades).
 * 4. Provides filtering options for candlestick data.
 * 5. Predicts future temperatures based on historical data.
 */

int main() {
    // Specify and Parse CSV File
    std::string filename = "weather_data.csv";
    auto data = readCSV(filename); ///< Parses the CSV file into a 2D
vector of strings.

    // Validate CSV Parsing
    if (data.empty()) {
        std::cerr << "Error: Failed to parse the CSV file or file is
empty.\n";
        return 1; // Exit with error code
    }

    // Display the first few rows for verification
    std::cout << "First few rows of the CSV file:\n";
    for (size_t i = 0; i < 5 && i < data.size(); ++i) {
        for (const auto& cell : data[i]) {
```

```
            std::cout << cell << " ";
        }
        std::cout << std::endl;
    }

    // --- Task 1: Candlestick Data Computation ---

    try {
        // Compute candlestick data for Austria ("AT") grouped by year
        std::cout << "\nComputing candlestick data for Austria (AT) by
year...\n";
        auto candlesticks = computeCandlestickData(data, "AT", "year");

        if (candlesticks.empty()) {
            std::cerr << "No candlestick data could be computed. Check
input data.\n";
            return 1;
        }

        // Display the computed candlestick data
        std::cout << "\nComputed Candlestick Data:\n";
        for (const auto& candle : candlesticks) {
            std::cout << "Date: " << candle.date
                      << ", Open: " << candle.open
                      << ", High: " << candle.high
                      << ", Low: " << candle.low
                      << ", Close: " << candle.close << std::endl;
        }

    // --- Task 2: Plot Candlestick Data ---

        // Plot the candlestick data grouped by decade
        std::cout << "\nText-Based Plot of Candlesticks for Austria
(AT) by Decade:\n";
        std::cout << "-----------------------------------\n";
        plotGroupedCandlesticks(candlesticks);

        // Main Menu for User Actions
        char proceed;
        do {
            std::cout << "\nChoose an option:\n";
            std::cout << "1. Filter and plot data (Task 3)\n";
            std::cout << "2. Predict temperatures (Task 4)\n";
```

```
            std::cout << "0. Exit\n";
            std::cout << "Enter your choice: ";
            int choice;
            std::cin >> choice;

    // --- Task 3: Filtering Options ---

            switch (choice) {
                case 1: {
                    std::cout << "\nWould you like to filter the data?
(y/n): ";
                    char filter_choice;
                    std::cin >> filter_choice;

                    if (filter_choice == 'y' || filter_choice == 'Y') {
                        std::cout << "\nChoose a filtering option:\n";
                        std::cout << "1. Filter by country\n";
                        std::cout << "2. Filter by date range\n";
                        std::cout << "3. Filter by temperature
range\n";
                        std::cout << "Enter your choice: ";
                        int filter_option;
                        std::cin >> filter_option;

                        std::vector<Candlestick> filtered_data;

                        switch (filter_option) {
                            case 1: {
                                // Display available countries
                                displayAvailableCountries(data);

                                // Filter by country
                                std::string country_prefix;
                                std::cout << "(Kindly input in
UPPERCASE)\n";
                                std::cout << "Enter the country prefix
(e.g., 'AT' for Austria):";
                                std::cin >> country_prefix;
                                filtered_data = filterByCountry(data,
country_prefix, "year");
                                break;
                            }
                            case 2: {
```

```
                                // Display available date range
                                displayAvailableDateRange(data);

                                // Filter by date range
                                std::string start_date, end_date;
                                std::cout << "Enter start date (YYYY):
";
                                std::cin >> start_date;
                                std::cout << "Enter end date (YYYY): ";
                                std::cin >> end_date;
                                filtered_data =
filterByDateRange(candlesticks, start_date, end_date);
                                break;
                            }
                            case 3: {
                                // Display available temperature range
                                displayAvailableTemperatureRange(data);

                                // Filter by temperature range
                                double min_temp, max_temp;
                                std::cout << "Enter minimum
temperature: ";
                                std::cin >> min_temp;
                                std::cout << "Enter maximum
temperature: ";
                                std::cin >> max_temp;

                                std::cout << "Filtering
candlesticks...\n";
                                filtered_data =
filterByTemperatureRange(candlesticks, min_temp, max_temp);
                                break;
                            }
                            default:
                                std::cerr << "Invalid choice. Exiting
filtering...\n";
                                return 1;
                        }

                        // Plot the filtered data
                        if (!filtered_data.empty()) {
                            std::cout << "\nFiltered and Plotted
Candlestick Data:\n";
```

```
                            plotGroupedCandlesticks(filtered_data);
                        } else {
                            std::cout << "No data available for the
selected filter.\n";
                        }
                    }
                    break;
                }

    // --- Task 4: Predictive Modelling ---

                case 2: {
                    // Predict temperatures
                    std::cout << "\nTask 4: Predicting Temperatures\n";
                    // Display available countries
                    displayAvailableCountries(data);

                    // Prompt user to select country
                    std::string country_prefix;
                    std::cout << "(Kindly input in UPPERCASE)\n";
                    std::cout << "Enter country prefix for prediction
(e.g., 'AT' for Austria):";
                    std::cin >> country_prefix;

                    // Prompt user for start and end years
                    int startYear, endYear;
                    std::cout << "Enter start year for prediction: ";
                    std::cin >> startYear;
                    std::cout << "Enter end year for prediction: ";
                    std::cin >> endYear;

                    // Perform prediction
                    predictAndDisplayTemperatures(data, country_prefix,
startYear, endYear);
                    break;
                }
                case 0:
                    std::cout << "Exiting program.\n";
                    proceed = 'n';
                    break;
                default:
                    std::cerr << "Invalid choice. Please try again.\n";
                    break;
```

```
            }

            if (choice != 0) {
                std::cout << "\nWould you like to perform another task?
(y/n): ";
                std::cin >> proceed;
            }
        } while (proceed == 'y' || proceed == 'Y');

    } catch (const std::exception& e) {
        // Handle any errors during computation or plotting
        std::cerr << "An error occurred: " << e.what() << std::endl;
        return 1; // Exit with error code
    }

    return 0; // Program Exit Successfully
}
```