```cpp
#ifndef UTILS_H
#define UTILS_H

#include <vector>
#include <string>
#include "Candlestick.h"
#include <map>

// --- General Utility Functions ---

/**
 * Reads a CSV file and returns its content as a 2D vector of strings.
 *
 * @param filename The name of the CSV file to read.
 * @return A 2D vector of strings, where each inner vector represents a
row of the file.
 */
std::vector<std::vector<std::string>> readCSV(const std::string
&filename);

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

/**
 * Computes candlestick data for a given country and time frame.
 *
 * @param data The dataset as a 2D vector of strings.
 * @param country_prefix The country prefix (e.g., "AT" for Austria).
 * @param time_frame The time frame (e.g., "year", "month", or "day").
 * @return A vector of computed Candlestick objects.
 */
std::vector<Candlestick> computeCandlestickData(
    const std::vector<std::vector<std::string>> &data,
    const std::string &country_prefix,
    const std::string &time_frame
);

// --- Task 2: Plotting Functions ---
/**
 * Plots candlestick data as a text-based graph.
 *
 * @param candlesticks A vector of Candlestick objects to plot.
 */
void plotCandlesticks(const std::vector<Candlestick>& candlesticks);
```

```cpp
/**
 * Creates a grouped text-based plot of candlestick data.
 *
 * @param candlesticks A vector of Candlestick objects to plot.
 * @param plot_height The height of the plot.
 */
void plotGroupedCandlesticks(const std::vector<Candlestick>&
candlesticks, int plot_height = 20);

// --- Task 3: Filtering Functions ---

/**
 * Filters candlestick data by a specified date range.
 *
 * @param candlesticks A vector of Candlestick objects to filter.
 * @param start_date The start date of the range.
 * @param end_date The end date of the range.
 * @return A vector of filtered Candlestick objects.
 */
std::vector<Candlestick> filterByDateRange(
    const std::vector<Candlestick>& candlesticks,
    const std::string& start_date,
    const std::string& end_date
);

/**
 * Filters candlestick data by a specified temperature range.
 *
 * @param candlesticks A vector of Candlestick objects to filter.
 * @param min_temp The minimum temperature.
 * @param max_temp The maximum temperature.
 * @return A vector of filtered Candlestick objects.
 */
std::vector<Candlestick> filterByTemperatureRange(
    const std::vector<Candlestick>& candlesticks,
    double min_temp,
    double max_temp
);

/**
 * Filters by a specific country and time frame.
 *
 * @param data The dataset as a 2D vector of strings.
 * @param country_prefix The country prefix (e.g., "AT" for Austria).
 * @param time_frame The time frame (e.g., "year", "month", or "day").
 * @return A vector of filtered Candlestick objects.
 */
std::vector<Candlestick> filterByCountry(
    const std::vector<std::vector<std::string>>& data,
    const std::string& country_prefix,
    const std::string& time_frame
);

// Display Filter Options
/**
 * Displays available countries in the dataset.
 *
 * @param data The dataset as a 2D vector of strings.
 */
void displayAvailableCountries(const
std::vector<std::vector<std::string>>& data);

/**
 * Displays the available date range in the dataset.
 *
 * @param data The dataset as a 2D vector of strings.
 */
void displayAvailableDateRange(const
std::vector<std::vector<std::string>>& data);

/**
 * Displays the available temperature range in the dataset.
 *
 * @param data The dataset as a 2D vector of strings.
 */
void displayAvailableTemperatureRange(const
std::vector<std::vector<std::string>>& data);

// --- Task 4: Polynomial Regression ---

/**
 * Performs polynomial regression to predict values.
 *
 * @param x A vector of x-values (e.g., years).
 * @param y A vector of y-values (e.g., temperatures).
```

```cpp
 * @param degree The degree of the polynomial to fit.
 * @param predict_x A vector of x-values for which predictions are
made.
 * @return A vector of predicted y-values.
 */
std::vector<double> polynomialRegression(
    const std::vector<int>& x,
    const std::vector<double>& y,
    int degree,
    const std::vector<int>& predict_x
);

/**
 * Predicts and displays temperature trends for a given country and
date range.
 *
 * @param data The dataset as a 2D vector of strings.
 * @param country_prefix The country prefix.
 * @param startYear The start year for the prediction.
 * @param endYear The end year for the prediction.
 */
void predictAndDisplayTemperatures(
    const std::vector<std::vector<std::string>>& data,
    const std::string& country_prefix,
    int startYear,
    int endYear
);

#endif // UTILS_H
```