Weather Data Logger

Specifications:

Variables: Date, temperature, humidity, and weather conditions.

Static & Const: Static variable for total records; const for maximum days.

Switch Case: Menu for logging, viewing, and analyzing weather data.

Looping Statements: Loop through weather records.

Pointers: Pointer for data manipulation.

Functions: Separate functions for logging, viewing, and analysis.

Arrays: Store weather data.

Structures: Structure for weather details.

Nested Structures: Nested structures for date and weather conditions.

Unions: Union for different weather metrics.

Nested Unions: Nested union for temperature and humidity details.

Output Expectations: Display weather data and analysis.

Menu Example:

1. Log Weather Data

2. View Weather Data

3. Analyze Weather Data

4. Exit

ANSWERS

```
#include <stdio.h>

#include <stdlib.h>

#define MAX_DAYS 30

struct Date {

    int day;

    int month;

    int year;
```

```
};

union WeatherMetrics {

    float temperature;

    float humidity;

};

struct WeatherConditions {

    char condition[20];

    union WeatherMetrics metrics;

};

struct WeatherData {

    struct Date date;

    struct WeatherConditions conditions;

};

void logWeatherData(struct WeatherData *data, int *count);

void viewWeatherData(struct WeatherData *data, int count);

void analyzeWeatherData(struct WeatherData *data, int count);

static int totalRecords = 0;

int main() {

    struct WeatherData weatherData[MAX_DAYS];

    int count = 0;

    int choice;

    while (1) {

        printf("\nWeather Data Logger Menu\n");

        printf("1. Log Weather Data\n");

        printf("2. View Weather Data\n");

        printf("3. Analyze Weather Data\n");

        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                logWeatherData(weatherData, &count);
                break;
            case 2:
                viewWeatherData(weatherData, count);
                break;
            case 3:
                analyzeWeatherData(weatherData, count);
                break;
            case 4:
                printf("Exiting program.\n");
                return 0;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }
}
void logWeatherData(struct WeatherData *data, int *count) {
    if (*count >= MAX_DAYS) {
        printf("Maximum days reached. Cannot log more data.\n");
        return;
    }
    printf("\nEnter weather details:\n");
    printf("Enter date (day month year): ");
    scanf("%d %d %d", &data[*count].date.day, &data[*count].date.month,
&data[*count].date.year);

    printf("Enter weather condition (e.g., Sunny, Rainy): ");

    scanf("%s", data[*count].conditions.condition);

    printf("Enter temperature (in Celsius): ");

    scanf("%f", &data[*count].conditions.metrics.temperature);

    totalRecords++;

    (*count)++;

    printf("Weather data logged successfully.\n");

}

void viewWeatherData(struct WeatherData *data, int count) {

    if (count == 0) {

        printf("No data available to view.\n");

        return;

    }

    printf("\nWeather Data:\n");

    for (int i = 0; i < count; i++) {

        printf("\nDay %d/%d/%d\n", data[i].date.day, data[i].date.month, data[i].date.year);

        printf("Weather Condition: %s\n", data[i].conditions.condition);

        printf("Temperature: %.2f°C\n", data[i].conditions.metrics.temperature);


    }

}

void analyzeWeatherData(struct WeatherData *data, int count) {

    if (count == 0) {

        printf("No data available for analysis.\n");

        return;

    }
```

```
     float totalTemp = 0;
    float maxTemp = data[0].conditions.metrics.temperature;
    float minTemp = data[0].conditions.metrics.temperature;
    for (int i = 0; i < count; i++) {
        totalTemp += data[i].conditions.metrics.temperature;
        if (data[i].conditions.metrics.temperature > maxTemp) {
            maxTemp = data[i].conditions.metrics.temperature;
        }
        if (data[i].conditions.metrics.temperature < minTemp) {
            minTemp = data[i].conditions.metrics.temperature;
        }
    }
    float averageTemp = totalTemp / count;
    printf("\nWeather Data Analysis:\n");
    printf("Average Temperature: %.2f°C\n", averageTemp);
    printf("Maximum Temperature: %.2f°C\n", maxTemp);
    printf("Minimum Temperature: %.2f°C\n", minTemp);
}
```