EECS348: Software Engineering C Programming Lab

Temperature Conversion Utility

Objective: Write a program in C that performs temperature conversions between Fahrenheit, Celsius, and Kelvin. The program will also categorize the temperature into predefined ranges based on its value and provide a weather advisory (e.g., "Freezing," "Comfortable," "Extreme Heat").

1. Input

The program will prompt the user to:

- Enter the temperature value.
- Choose the temperature scale of the input value (Fahrenheit, Celsius, or Kelvin).
- Choose the conversion target (Fahrenheit, Celsius, or Kelvin).

2. Output

The program will:

- Convert the temperature based on the user’s input scale and desired target scale.
- Display the converted temperature.
- Categorize the converted temperature into one of the following categories:
  - Freezing: Below 0°C (or equivalent in other scales)
  - Cold: 0°C to 10°C (or equivalent)
  - Comfortable: 10°C to 25°C (or equivalent)
  - Hot: 25°C to 35°C (or equivalent)
  - Extreme Heat: Above 35°C (or equivalent)
- Provide a simple weather advisory based on the category (e.g., "Wear a jacket," "Stay indoors").

Requirements: Implement at least four functions:

1. `float celsius_to_fahrenheit (float celsius)`
   Converts Celsius to Fahrenheit using the formula: $F = (9/5)C + 32$

2. `float fahrenheit_to_celsius (float fahrenheit)`
   Converts Fahrenheit to Celsius using the formula: $C = (5/9)(F - 32)$

3. `float celsius_to_kelvin (float celsius)`
   Converts Celsius to Kelvin using the formula: $K = C + 273.15$

4. `float kelvin_to_celsius (float kelvin)`
   Converts Kelvin to Celsius using the formula: $C = K - 273.15$

Add any additional functions as necessary for conversions between Fahrenheit and Kelvin.

Categorization and Advisory Function

Write a function that categorizes the temperature into “Freezing,” “Cold,” “Comfortable,” “Hot,” or “Extreme Heat” based on the Celsius value, and provides an advisory message.

```
void categorize_temperature(float celsius)
```

Error Handling and Input Validation

The program should handle invalid inputs gracefully, including:
- Invalid temperature values (e.g., negative Kelvin values).
- Invalid conversion choices (e.g., converting from Kelvin to Kelvin).

Additional Notes:
- Input Validation: Ensure the user inputs valid numbers and correct choices for conversion.
- Edge Case Handling: Consider extreme temperatures and invalid cases like negative Kelvin values.
- Error Messages: Display clear error messages for invalid inputs and provide an option to re-enter values.

Grading Criteria

- Correctness: Program produces accurate results for all input cases. [20]
- Structure: Functions are well-organized and properly used. [10]
- Validation: Program gracefully handles invalid input. [10]
- Documentation: Code includes comments explaining logic and flow. [10]

Example Run

```
Example 1:
Enter the temperature: 300
Choose the current scale (1) Celsius, (2) Fahrenheit, (3) Kelvin: 3
Convert to (1) Celsius, (2) Fahrenheit, (3) Kelvin: 1

Converted temperature: 26.85°C
Temperature category: Comfortable
Weather advisory: You should feel comfortable.

Example 2:
Enter the temperature: 20
Choose the current scale (1) Celsius, (2) Fahrenheit, (3) Kelvin: 2
Convert to (1) Celsius, (2) Fahrenheit, (3) Kelvin: 3

Converted temperature: 266.48K
Temperature category: Cold
Weather advisory: Wear a jacket.
```