MTH 4300: Algorithms, Computers and Programming II

Fall 2025

Section: STRA

Problem Set 1

Due Date: September 4, 2025

Question 1: Temperature and Distance Converter

Write a C++ program that performs unit conversions and calculations using the concepts from Lecture 2:

1. **Input Collection:**
   - Ask for the user's name (handle spaces properly using `std::getline()`)
   - Ask for a temperature in Fahrenheit (can be decimal)
   - Ask for a distance in miles (can be decimal)
   - Ask for number of hours traveled (integer)
2. **Conversions and Calculations:**
   - Convert Fahrenheit to Celsius using: $C = (F - 32) \times 5/9$
   - Convert miles to kilometers using: $km = miles \times 1.60934$
   - Calculate average speed in miles per hour: $speed = distance \div hours$
   - Convert that speed to kilometers per hour
3. **Output Display:**
   - Show the user's name
   - Display temperature in both Fahrenheit and Celsius (2 decimal places)
   - Display distance in both miles and kilometers (2 decimal places)
   - Show average speed in both mph and km/h (2 decimal places)

Requirements:

- Use appropriate data types (`std::string`, `double`, `int`)
- Use `std::getline()` for name input
- Use `static_cast<double>()` when dividing integers to ensure proper decimal results
- Use compound assignment operators where appropriate (e.g., for intermediate calculations)
- Format all decimal outputs to 2 decimal places using `std::fixed` and `std::setprecision()`
- Include proper headers (`#include <iostream>`, `#include <string>`, `#include <iomanip>`)

Provide complete source code and sample output demonstrating all conversions and calculations.

Sample Output:

```
=== Temperature and Distance Converter ===

Enter your name: Sarah Johnson
Enter temperature in Fahrenheit: 75.5
Enter distance in miles: 120.75
Enter hours traveled: 3

=== Conversion Results ===
Name: Sarah Johnson
Temperature: 75.50°F = 24.17°C
Distance: 120.75 miles = 194.33 km
Average Speed: 40.25 mph = 64.78 km/h
```

Solution:

```cpp
#include <iostream>
#include <string>
#include <iomanip>

int main() {
    std::string name;
    double fahrenheit, miles;
    int hours;

    std::cout << "=== Temperature and Distance Converter ===" << std::endl << std::endl;

    // Input collection
    std::cout << "Enter your name: ";
    std::getline(std::cin, name);

    std::cout << "Enter temperature in Fahrenheit: ";
    std::cin >> fahrenheit;

    std::cout << "Enter distance in miles: ";
    std::cin >> miles;

    std::cout << "Enter hours traveled: ";
    std::cin >> hours;

    // Conversions and calculations
    double celsius = (fahrenheit - 32) * 5.0 / 9.0;
    double kilometers = miles * 1.60934;
    double speed_mph = miles / static_cast<double>(hours);
    double speed_kmh = speed_mph * 1.60934;

    // Output display
    std::cout << std::endl << "=== Conversion Results ===" << std::endl;
    std::cout << "Name: " << name << std::endl;
    std::cout << std::fixed << std::setprecision(2);
    std::cout << "Temperature: " << fahrenheit << "°F = " << celsius << "°C" << std::endl;
    std::cout << "Distance: " << miles << " miles = " << kilometers << " km" << std::endl;
    std::cout << "Average Speed: " << speed_mph << " mph = " << speed_kmh << " km/h" << std::endl;

    return 0;
}
```

Question 2: Recipe Scaling Calculator

Write a C++ program that helps scale a recipe up or down based on the number of servings needed. Your program should demonstrate arithmetic operations, type casting, and I/O operations:

1. **Input Collection:**
   - Ask for the recipe name (handle spaces using `std::getline()`)
   - Ask for original number of servings the recipe makes (integer)
   - Ask for desired number of servings (integer)
   - Ask for three ingredient amounts from the original recipe:
     - Flour in cups (can be decimal, like 2.5)
     - Sugar in tablespoons (integer)
     - Milk in fluid ounces (can be decimal)
2. **Scaling Calculations:**
   - Calculate the scaling factor: desired servings ÷ original servings
   - Scale each ingredient by multiplying by the scaling factor
   - Convert scaled flour from cups to tablespoons (1 cup = 16 tablespoons)
   - Convert scaled milk from fluid ounces to cups (1 cup = 8 fluid ounces)
3. **Demonstration of Operators:**
   - Use compound assignment operators to adjust one ingredient amount (e.g., add 0.25 cups to flour using `+=`)
   - Show increment operators by increasing the serving count: `int new_servings = desired_servings++`
   - Calculate total dry ingredients: flour (in tbsp) + sugar (in tbsp)
4. **Output Display:**
   - Show recipe name and scaling information
   - Display original and scaled ingredient amounts
   - Show converted measurements (flour in tbsp, milk in cups)
   - Display the final serving count after increment operation

Requirements:
- Use appropriate data types (`std::string`, `int`, `double`)
- Use `static_cast<double>()` when dividing integers for the scaling factor
- Use compound assignment and increment operators as specified
- Format decimal outputs to 2 decimal places
- Include headers: `#include <iostream>`, `#include <string>`, `#include <iomanip>`

Provide complete source code and sample output showing the recipe scaling process.

Sample Output:

```
=== Recipe Scaling Calculator ===

Enter recipe name: Chocolate Chip Cookies
Enter original number of servings: 12
Enter desired number of servings: 18
Enter flour amount (cups): 2.5
Enter sugar amount (tablespoons): 8
Enter milk amount (fluid ounces): 6.0

=== Scaling Results ===
Recipe: Chocolate Chip Cookies
Scaling factor: 1.50

Original Ingredients:
- Flour: 2.50 cups
- Sugar: 8 tablespoons
- Milk: 6.00 fluid ounces

Scaled Ingredients:
- Flour: 3.75 cups (60.00 tablespoons)
- Sugar: 12.00 tablespoons
- Milk: 9.00 fluid ounces (1.12 cups)

After adjustments:
- Flour (adjusted): 4.00 cups
- Total dry ingredients: 72.00 tablespoons
- Final serving count: 19
```

Solution:

```cpp
#include <iostream>
#include <string>
#include <iomanip>

int main() {
    std::string recipe_name;
    int original_servings, desired_servings;
    double flour_cups, milk_oz;
    int sugar_tbsp;

    std::cout << "=== Recipe Scaling Calculator ===" << std::endl << std::endl;

    // Input collection
    std::cout << "Enter recipe name: ";
    std::getline(std::cin, recipe_name);

    std::cout << "Enter original number of servings: ";
    std::cin >> original_servings;

    std::cout << "Enter desired number of servings: ";
    std::cin >> desired_servings;

    std::cout << "Enter flour amount (cups): ";
    std::cin >> flour_cups;

    std::cout << "Enter sugar amount (tablespoons): ";
    std::cin >> sugar_tbsp;

    std::cout << "Enter milk amount (fluid ounces): ";
    std::cin >> milk_oz;

    // Scaling calculations
    double scaling_factor = static_cast<double>(desired_servings) / static_cast<double>(original_servings);

    double scaled_flour = flour_cups * scaling_factor;
    double scaled_sugar = static_cast<double>(sugar_tbsp) * scaling_factor;
    double scaled_milk = milk_oz * scaling_factor;

    // Conversions
    double flour_tbsp = scaled_flour * 16.0;
    double milk_cups = scaled_milk / 8.0;

    // Demonstration of operators
    scaled_flour += 0.25; // Compound assignment
    double adjusted_flour_tbsp = scaled_flour * 16.0; // Recalculate after adjustment

    int new_servings = desired_servings++; // Post-increment

    // Total dry ingredients using original scaled amounts (before adjustment)
    double total_dry_tbsp = flour_tbsp + scaled_sugar;

    // Output display
    std::cout << std::endl << "=== Scaling Results ===" << std::endl;
    std::cout << "Recipe: " << recipe_name << std::endl;
    std::cout << std::fixed << std::setprecision(2);
    std::cout << "Scaling factor: " << scaling_factor << std::endl << std::endl;

    std::cout << "Original Ingredients:" << std::endl;
    std::cout << "- Flour: " << flour_cups << " cups" << std::endl;
    std::cout << "- Sugar: " << sugar_tbsp << " tablespoons" << std::endl;
    std::cout << "- Milk: " << milk_oz << " fluid ounces" << std::endl << std::endl;

    std::cout << "Scaled Ingredients:" << std::endl;
    std::cout << "- Flour: " << flour_cups * scaling_factor << " cups (" << flour_tbsp << " tablespoons)" <<
std::endl;
    std::cout << "- Sugar: " << scaled_sugar << " tablespoons" << std::endl;
    std::cout << "- Milk: " << scaled_milk << " fluid ounces (" << milk_cups << " cups)" << std::endl << std::endl;

    std::cout << "After adjustments:" << std::endl;
    std::cout << "- Flour (adjusted): " << scaled_flour << " cups" << std::endl;
    std::cout << "- Total dry ingredients: " << total_dry_tbsp << " tablespoons" << std::endl;
    std::cout << "- Final serving count: " << desired_servings << std::endl;

    return 0;
}
```