Midterm exam

CSE241

April 21, 2022

This part of the exam is in **two (2)** parts. You must complete both parts. Save each part in a separate `.cpp` file. Each `.cpp` should have its own, separate, `main` function. Submit both `.cpp files` (optionally in a `.zip` file, if you choose) before 12:00pm.

Makefiles are not required.

1 Weather processing

1.1 Description

You are to write a data processing program in C++, specifically processing weather data. The program should read in the data on standard input.

The data is formatted with one `double` per line. Each number represents one day's temperature, in degrees Celsius.

If there is bad input (a number cannot be read in), ignore that line and continue on to the next line.

You may assume that the number of lines will never exceed 20.

Your program must output:

- The total number of lines read in (even invalid ones)
- The total number of *valid* records read in
- The mean temperature (with one (1) digit after the decimal point)
- A formatted table of all temperatures (right-aligned, with three (3) digits after the decimal point) which were colder than than mean

1.2 Sample input and expected output

Download the input file `midterm-exam-weather.in` for testing. Output should be identical to:

```
9 lines read in
6 valid lines read in
mean temperature is 21.2
temperatures colder than the mean:
    20.000
    10.500
    19.210
    21.175
```

2 Divisible by 3

2.1 Description

Complete a C++ program which determines if a string of digits represents an integer which is divisible by 3. Note: you should *not* attempt to read in or parse as an integer, as the number may be too large to fit into any C++ integer type. All of your calculations should be done on strings.

You may assume the integer represented by the string is never negative and never invalid. I.e., it is not required to check for bad input.

A number is divisible by 3 if the summation of its digits is divisible by 3.

To receive full marks for this part, you *must* complete and make use of the following *recursive* function:

```
char sum_digits(char *);
```

This function will read in, as a string, a sequence of digits. It will then return a single digit, represented as a character. E.g., if the input is “`128`”, you must return ‘2’ (because $1 + 2 + 8 = 11$ and $1 + 1 = 2$).

Use the following logic to write your recursive function:

- If the string is empty, its digits sum to 0
- If a string consists of only a single digit, then return that digit
- If a string begins with the digit `0`, then you can ignore that `0` and recurse on the rest of the string
- Otherwise, add the first two digits together. E.g., if the first two digits are `3` and `9`, add them to get 12. Put the `1` (the first digit of 12) in the first position of the string and the `2` (second digit of 12) in the second position of the string and recurse.

Here is an example of calculating for the string `172931`:

$$\begin{aligned}
& \textit{sum\_digits}(\text{“}172931'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}082931'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}82931'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}10931'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}01931'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}1931'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}1031'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}0131'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}131'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}041'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}41'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}05'') \\
\Rightarrow\ & \textit{sum\_digits}(\text{“}5'') \\
\Rightarrow\ & \text{“}5''
\end{aligned}$$

To convert between integer representations and character representations:

```
char x = '3';
int y = x - '0';  // y will have the integer value 3
char z = y + '0'; // z will have the character value '3'
```

You may make use of the attached `main` function (see `midterm-part2.cpp`) so that you do not have to write the main function or do testing yourself.