CSC 116
Fundamentals of Programming with Engineering Applications II
Assignment 2

Changelog: 2020-09-26. What to do if filename in command line does not exist.
Changelog: 2020-09-25. Fixed errata in description of Ada's numbers.

Note 1 **This assignment is to be done individually**
Note 2 You can discuss the assignment with others, but copying code is prohibited.

A note on Academic Integrity and Plagiarim

Please review the following documents:

- Standards for Professional Behaviour, Faculty of Engineering: `https://www.uvic.ca/engineering/assets/docs/professional-behaviour.pdf`
- Policies Academic Integrity, UVic: `https://www.uvic.ca/students/academics/academic-integrity/`
- Uvic's Calendar section on Plagirism: `https://www.uvic.ca/calendar/undergrad/index.php#/policy/Sk_0xsM_V`

Note specifically:

> Plagiarism
> Single or multiple instances of inadequate attribution of sources should result in a failing grade for the work. A largely or fully plagiarized piece of work should result in a grade of F for the course.

A program you submit will be considered a **piece of work**. You are responsible for your own submission, but you could also be responsible if somebody plagiarizes your submission.

Objectives

After completing this assignment, you will have experience with:

- Basic usage of strings
- Basic usage of vectors
- Basic use of structs
- Basic input

Your task, should you choose to accept it

What is the name of the baby? This is a typical question that new parents hear over and over again. For this assignment we will create a program to help future parents (and relatives) choose a name, using data on baby names from the provincial government.

Names of babies

The BC government makes available, for each baby name, the frequency that it has been used in a particular year. This data spans 100 years and it is divided by gender (*boys* and *girls*)[1]. We have pre-processed the data, so it is easy to read into a C++ program. We have provided one input file, `babies.txt`, that combines these two files into one. The format of this file is the following:

1. If, in one year, there are less than 5 babies of the same gender with the same name, the dataset reports a zero for that year.

- Each line corresponds to a baby.
- Each field in the line is separated by a comma: `,`
- Each line has exactly 102 fields
- The first field is the name of the baby
- The next 100 fields are a list of frequencies. Each frequency corresponds to a year. The first element is the frequency for 1920, the second for 1921, etc. The 100th element is the frequency for 2019.
- The last element is the sum of the frequencies. Its purpose is to verify the consistency of the data.

An example is shown below. Note that it is one single line in the input file. It has been wrapped due to the limitations of the width of the page. In this example the baby's name is *ADA* (names are always in uppercase characters), in 1920 10 had that name, 0 in 1921, 12 in 1922, ..., 20 in 2018, 23 in 2019. There were a total of 300 babies named *ADA* in BC in the last 100 years.

```
ADA,10,0,12,0,8,0,7,0,6,0,7,6,5,7,8,0,0,0,0,0,6,5,0,0,0,0,6,0,0,5,6,0,0,0,0,0,0,0,5,0,
0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,
0,0,7,0,8,9,8,11,10,12,12,14,14,19,9,20,23,300
```

Code provided

Download the source code from conneX.

Specification

- The program `babies.cpp` takes one command line parameter: the data file. We are give one data file: `babies.txt` (in the directory data). If the file does not exist, the program should terminate with the following error and exit code 1.
  `Unable to open input file.`
- The program should be capable of reading as many baby records are there are in the file (and you should not assume that the number of records is fixed).
- When all of the records have been processed, the program will print the total number of records read.
- After processing records, the program will read names from standard input (one name per line), and output a summary of the statistics for that name (see below for the exact format of the summary). If the entered name does not exist, the program will print an error message, but continue reading names to look up.
- The program should be capable of reading and looking up as many baby names as necessary (even when some do not exist).

For example, when run as follows:

```
./babies data/unix/babies.txt
```

with input

```
YODA
ADA
```

The output should be

```
Read 4340 babies.
YODA
Baby name [YODA] was not found
ADA
 Total: 300
 Years: 32
```

```
2019: 23
First: 1920 10
Last: 2019 23
Min: 1932 5
Max: 2019 23
Avg: 3
```

Note that in the output above, the each of the lines containing a statistic value begins with a space. Your program is expected to duplicate this behavior.

Your code

Modify the code provided in `babies.cpp`. You only need to implement the following functions. You should leave the rest of the code unchanged.

1. `Read_Babies`

Given an filename, read all the input lines into a vector of `baby_type` struct and return this vector. If the file does not exist, it should print the error message `Unable to open input file.` and terminate with a exit code of 1 (use `exit(1);`).

2. `Print_Baby`

Given a `baby_type` struct, print the following information:

- **Name**
- **Total**: total number of babies (with that name).
- **Years**: number of years with at least one baby.
- **2019**: count for the year 2019.
- **First**: first year (and count) with at least one baby.
- **Last**: last year (and count) with at least one baby.
- **Min**: year (and count) of the minimum number of babies (ignore years with zero babies). If two or more years have the same count, print the **first** year with the minimum number.
- **Max**: year (and count) of the maximum number of babies. If two or more years have the same count, print the **last** year with the maximum number.
- **Avg**: Average number (divide count by number of years–100).

Other than the first line containing the name, each of the lines should be preceded by a single space character.

3. `Find_Baby`

This function receives three parameters: the vector `name_list` containing `baby_type` structs, a `name` to look up, and a reference `output_baby` to the location where the corresponding record should be placed. If `name` of the baby exists in the vector, the function will copy his/her record to `output_baby` and return `true`. If the baby does not exist, the function will return `false` without modifying `output_baby`. Names should be identical; hence `Yoda` should be considered to be different from `YODA`.

Evaluation

Solutions should be:

1. Correct. They should pass all the tests we have provided.
2. In good style, including indentation and line breaks

What to submit

- Using conneX, submit your version of the file `babies.cpp`. **Do not submit any other files**.
- You should assume that your submission was received correctly only if both of the following two conditions hold.
  - You receive a confirmation email from conneX.
  - You can download and view your submission, and it contains the correct data.
- In the event that your submission is not received by the marker, you will need to demonstrate that both of the above conditions were met if you want to argue that there was a submission error.