CSC 330
Programming Languages
Assignment 1

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

A note on Academic Integrity and Plagiarism

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.

Submissions will be screened for plagiarism **at the end of the term**.

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:

1. Familiarize with SML
2. Basic usage of recursion
3. Basic usage of tuples and/or records
4. Learn to program without mutation

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 SML program. We have provided one input file, `babies.txt`, that combines these two files into one. First, a data file correspond to a starting year (for example, 1920) which we will call $startYear$. 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.

1. Each line corresponds to a baby.
2. Each field in the line is separated by a comma: `,`
3. The first field is the name of the baby
4. The name of the baby is followed by n fields (all integers $>=$). n is guaranteed to be larger than 0.
5. The next n-1 fields are a list of frequencies (integers). Each frequency corresponds to a year. The i^{th} element (zero based) in the list corresponds to the frequency of the year $startYear + i$
6. The n^{th} element is the sum of the frequencies. Its purpose is to verify the consistency of the data.

Furthermore:

1. All records in the file have the same length and are all valid.
2. The sum is guaranteed to be the sum of the values for the years.

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 this page. This record has a $startYear = 1920$ 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,
0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0,
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 brightspace. Your program will be tested in `linux.csc.uvic.ca`. The source code, Makefile and tests assume this.

Specification

1. There are two source code files. You are expected to modify only `babies.sml` (you will only submit this file).
2. `csc330.sml` contains some functions that you will need to use. **Do not modify this file.** You will need several of the functions in this file to complete this assignment, including:
   (a) `split_at`: split given string at a given character, returns a list of strings. You can use this function to split the input into records and each record into fields.
   (b) several functions to convert from string to numeric values and vice-versa.
3. The program:
   - reads all the baby records in the input file;
   - reads names from standard input (one name per line)
   - prints the total number of records read;
   - for each name in the input: prints a summary of the statistics for that each name (see below) or an error if the name does not exist
4. The program is run with two parameters: the data file and the $startYear$. You are given one data file: `data/babies.txt` (other data files might be used during testing).
5. You are expected to write the function `babies_program`. This function takes 3 parameters:
   - The contents of the babies file, in one single string.
   - A string with a list of names of babies to search. One baby per line.
   - A string containing $startYear$ (the year of the first record in the babies file)

   This function returns a string that is then printed to the standard output. **This function should be pure**.

For example, when run as follows:

```
sml @SMLload  babies-image.x86-linux data/babies.txt 1920
```

with input

```
YODA
ADA
```

The output should be

```
Read 4340 babies. Starting year 1920. Each baby has 100 entries.
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, each of the lines containing a statistic value begins with a space. Your program is expected to duplicate this behavior.

- Total is the total number of babies in the period.
- Years is the number of years non-zero
- 2019, in this case, it is the year corresponding to the last year in the data. Remember that the first int of the list of years is the *offset* year.
- First is the year and value of the first non-zero value.
- Last is the year and value of the last non-zero value.
- Min is the year and value of the **first** minimum count (non zero)
- Max is the year and value of the **last** maximum count (non zero)

Your code

Modify the code provided in `babies.sml`. You only need to implement the function `babies_program`. You should leave the other files unchanged.

Makefile

I have provided a Makefile that shows how the binary is created and how to run it. Modify this Makefile to do your testing.

Restrictions

Your solution should satisfy the following restrictions:

1. Your program should generate **NO** warnings. If your program generates a single compilation warning it will receive a zero.
2. Your program should not make any assumptions regarding the number of years per record. However, all records will have the same number of years.
3. **It cannot use mutation**. Use of `ref`, `:=` and `!` is forbidden.
4. It **cannot use the `case`** expression.

5. It **cannot** use any library function that requires the explicit name of its structure when called (e.g. `TextIO.inputAll`, `List.nth`, `List.filter`, etc.). For this reason you are not allowed to use the character **.** in an identifier in your program. Your program will receive a zero if you do. You cannot circumvent this rule using `open`. Several functions of List can be used, because they do not require the name of its package, such as `hd`, `null`, `length`, etc.
6. It **cannot** use any of the following functions from the List structure (several other functions in this structure are forbidden due to the previous rule):
   (a) `foldl`, `foldr`, nor `map`.

Any program that violates a restriction will receive a zero.

Testing

You **must** test your program. I have provided 3 tests. They are in the test directory. See Makefile.

The output of your program should be **identical** to the expected output (see the tests directory).

Evaluation

Solutions should:

1. Compile and run in `linux.csc.uvic.ca` If a program does not compile **for any reason** it will receive a zero.
2. Satisfy the restrictions above.
3. Pass tests. Your grade will be determined based on how many tests it passes.
   (a) For each test, the output of your program will be compared using `diff`. `diff` should report that the output of your program is **identical** to the expected one. Otherwise the test has failed. See the `tests` directory for sample input and output.
4. Be correct. They should pass all the tests we have provided.
5. Be in good style, including indentation and line breaks

Second submission

If you choose, you can submit this assignment a second time (the deadline will be set after the results of the first submission are published). The final grade of your assignment will be:

$$max(first\_submission, 0.5 * second\_submission)$$

You are not required to submit a second time. In that case, your final grade will be the grade of the first submission.

Hints

1. The simplest test your program can pass (which does not require to complete the assignment) is when the list of names to process is empty. Thus you don't have to print any records.
2. Use recursion. You will need to use it to process the lists.
3. Use the function provided (`split_at`) to split the input file and the input from standard input.
4. Store each baby's information as a tuple (of 3 items, name, total, and list of year counts) or a record. Using a record will make your programming easier.
5. The program is not difficult, but you will have to learn to deal with compilation errors. I recommend:

    (a) Declare the types of your functions (parameters and return value)
    (b) Use records to store the babies
    (c) To determine if a list is empty use the function `null`
    (d) Use the REPL to test your assumptions about how SML works
6. Write your own tests.
7. Automatize your tests.
8. As far as I know, Emacs is the only editor that supports SML.

What to submit

1. Using brightspace, submit your version of the file `babies.sml`. **Do not submit any other files**.