COMP2401 - Assignment #6

(Due: **Wednesday, April 10th, 2024 @ 11pm**)

In this assignment, you will practice reading and writing information to/from both text & binary files.

We will be writing code that reads in a file of records with data about employees in a fictional company. The data was obtained from: (https://www.thespreadsheetguru.com/blog/sample-data)

In this assignment, we are just interested in reading in the text file, extracting some relevant data, re-writing it to a binary file and then re-reading that data and showing some statistics.

There are **1001** lines of data in the file, the first being a header that describes each field of data. There are **14** *pieces of data* on each line. We are only interested in **11** of those pieces of data (shown in red below). For a single line from the file, each of the **14** data values are separated by a **TAB** character (i.e., `'\t'`) and the line ends with a `'\n'` character.

- **Employee ID** – employee number (format: E00000)
- **Full Name** – employee's first and last name.
- **Job Title** – string representing job title of employee (see **company.h**).
- **Department** – string representing department of employee (see **company.h**).
- **Business Unit** – string representing business unit of employee (see **company.h**).
- **Gender** – ignore
- **Ethnicity** – ignore
- **Age** – integer representing the person's age
- **Hire Date** – date employee was hired (format: M/D/YYYY), M & D may be 1 or 2 chars
- **Annual Salary** – integer (format: $xxx,xxx)
- **Bonus %** – ignore
- **Country** – country employee works in
- **City** – city employee works in
- **Exit Date** – date employee left company (format: M/D/YYYYY), M & D may be 1 or 2 chars

HELPFUL TIP: In this assignment, you will need to read a text file and output a binary one. It would be a VERY good idea to get all the code written to read in the text file first and display all the values to the screen. Once this is working correctly, you can then add the code to write out the binary file.

Part 1

Download the file called **EmployeeSampleData.txt**. This is a text file with "tab-separated values". That means, a tab (i.e., `'\t'`) character separates the values on each line. It contains a header line, followed by **1,000** employee records. Write a program called **reduceData.c** that will read in this file, extract some data from it and then re-write it to a new file called **EmployeeDataReduced.bin**. You will also want to download the **company.h** file, which you can make use of on this assignment.

Here is what you need to do:

1. Make sure that you open the **.txt** file for reading and close it when you are done. You must also check for errors in opening the file and use **printf** to indicate if an error has occurred.

2. You must open the binary file for writing and close it when you are done. You must check for errors as well in case the file cannot be created. Each time you run the code, the binary file must be overwritten.

3. The program must read in all the records (i.e., lines) until none remain. Even though there are only 1,000 records (i.e., lines) … your code must work for ANY number of lines. Therefore, you are NOT allowed to hardcode the number of lines into your program. You may want to read the whole file first to determine the number of lines in it. Thus, your program should work properly (without having to re-compile) regardless of the number of lines in the original **.txt** file.

4. The binary file will contain information in the same order as the **.txt** file, however some of the **.txt** file data will be ignored and not written to the binary file. Also, some of the data will be re-formatted to be more compact to reduce file size.

5. You may want to read-in one piece of data at a time. Each piece of data is separated by a tab character. If you want to use **fscanf()**, you can make use of the following format string which will read everything up to (**but not including**) a tab character: `"%[^\t]s"`. And the following will read everything up to (**but not including**) a newline character: `"%[^\n]s"` … which is good for the last value in a line. So, to be clear, after using **fscanf()** like this, you will still need to read in the tab or newline character before going to the next field.

6. Also, you MUST output the data (into the binary file) as explained below. First output the number of records in the file as an **unsigned short**. Then output ONLY the following data to the binary file:

    - A **2-byte short integer** representing the employee number without the E
    - A **byte** indicating the number of characters in the employee's name
    - Each **character** from the employee's name (including the space character)
    - A single **byte** indicating the job title which is the index of the job title in the **JobTitle** array (see **company.h**). In the data, there is a special case to handle since the job of "System Administrator" actually has an extra character at the end of the string. You must handle this, but you cannot modify the text file.
    - A single **byte** indicating the department which is the index of the department in the **Departments** array (see **company.h**)
    - A single **byte** indicating the business unit which is the index of the business unit in the **BusinessUnits** array (see **company.h**).
    - A single **byte** indicating the age of the employee
    - A **2-byte short integer** indicating the year that the employee was hired
    - A **byte** indicating the month that the employee was hired
    - A **byte** indicating the day of the month that the employee was hired
    - An 4-byte **integer** indicating the salary of the employee

- A single **byte** indicating the country which is the index of the country in the **Countries** array (see **company.h**)
- A single **byte** indicating the city which is the index of the city in the **Cities** array (see **company.h**)
- A **2-byte short integer** indicating the year that the employee left the company (use the value 0 if the employee still works there). If the employee has left the company, then also output a single **byte** indicating the month that the employee left and a single **byte** indicating the day of the month that the employee left. If the employee is still working there, do not output these month and day values.

The original **EmployeeSampleData.txt** file is exactly **122,122** bytes in size. The final **EmployeeDataReduced.bin** file should be **31,326** bytes in size. You can use **ls -l** to confirm the file sizes. If you cannot get this exact number, leave it for now and go on to **Part 2**. Then if you get everything done, come back and see if you can adjust your code to get the correct number.

Part 2

Write a program called **stats.c** that reads in the **EmployeeDataReduced.bin** file that you just created and prints out the following information formatted in a reasonable way that explains the output:

- Determine the employee with the highest salary and the lowest salary and show their name, job title, salary, city and country. Also show the average salary for all current employees (i.e., average should not include employees that have exited the company). You can use the flags **%'d** in a `printf()` to allow the commas to be displayed in the monetary values. However, you will need to include `<locale.h>` and also call `setlocale(LC_NUMERIC, "");` one time in your code.
- Display a list (in any order) of all System Administrators from Seattle. Show just their employee ID and name.
- Display all employees hired in October of 2022, showing their employee ID, name and date of hire. They should be displayed in sorted order of date hired (longest working shown first).
- Display a list (sorted in increasing alphabetical order by name) for all employees in China who work in the IT department and in Research & Development units. Show just their employee ID and name.
- Display a list (in any order) for all employees in who are 65 years of age or older. Show just their employee ID, name and salary.
- Show how much money will be saved annually if those 65 or older exit the company.

Here is a sample output of what you should see.

```
Lowest Salary Employee
  $40,352 for Melody Yoon (Business Partner - Beijing,China)

Highest Salary Employee
  $258,734 for Robert Rogers (Vice President - Seattle,United States)

Average Salary of Current Employees = $109,843

System Administrators From Seattle:
-----------------------------------
  #E02010 - Gianna Holmes
  #E02461 - Delilah Alvarez
  #E02500 - Leilani Hong
  #E02754 - Kai Green
  #E02757 - Levi Rahman
```

```
Employees Hired in October of 2022:
-----------------------------------
  #E02780 - Samuel Patterson     (2022-10-07)
  #E02233 - Greyson Lim          (2022-10-12)
  #E02967 - Ava Chan             (2022-10-12)
  #E02516 - Jayden Phillips      (2022-10-13)
  #E02388 - Alexander Zhu        (2022-10-17)
  #E02318 - Carter Luu           (2022-10-20)
  #E02635 - Luna Chang           (2022-10-20)
  #E02837 - Natalie Thao         (2022-10-23)
  #E02789 - Everly Hwang         (2022-10-27)

IT Employees in China working in Research & Development:
--------------------------------------------------------
  #E02215 - Adrian Ngo
  #E02004 - Cameron Lo
  #E02439 - Hadley Le
  #E02581 - Hudson Oh
  #E02154 - Hunter Yoon
  #E02219 - Iris Chung
  #E02956 - Jaxon Lai
  #E02863 - Jose Park
  #E02097 - Joseph Tan
  #E02552 - Kai Duong
  #E02192 - Lillian Vang
  #E02969 - Lydia Chu
  #E02304 - Madison Xu
  #E02148 - Naomi Lee
  #E02726 - Samantha Do
  #E02173 - Zoey Leung

Employees 65 or older:
----------------------
  #E02639 - Asher Ly                $ 63,853
  #E02887 - Caroline Gomez          $ 90,737
  #E02483 - Caroline Nunez          $224,872
  #E02484 - Carson Brown            $149,474
  #E02117 - Chloe Yoon              $ 83,854
  #E02203 - Ellie Wilson            $164,102
  #E02696 - Everleigh Kumar         $125,213
  #E02326 - Gabriel Ahmed           $ 73,996
  #E02559 - Jackson Gupta           $ 77,065
  #E02754 - Kai Green               $ 75,439
  #E02055 - Layla Bell              $203,030
  #E02355 - Leonardo Li             $155,716
  #E02588 - Liam Baker              $ 97,379
  #E02764 - Lincoln Alvarado        $ 93,857
  #E02413 - Natalia Cheng           $ 47,919
  #E02828 - Nevaeh Jiang            $ 96,897
  #E02075 - Peyton Edwards          $ 59,344
  #E02781 - Robert Padilla          $ 63,346
  #E02435 - William Juarez          $ 88,533

$2,034,626 will be saved annually if those 65 or older exit the company
```

IMPORTANT SUBMISSION INSTRUCTIONS:

Submit:

1. A **Readme** text file containing
   - your name and studentNumber
   - a list of source files submitted
   - any specific instructions for compiling and/or running your code
2. All of your **.c source** files and all other files needed for testing/running your programs, including the original **.txt** file.

The code **MUST** compile and run on the course VM.

- If your internet connection at home is down or does not work, we will not accept this as a reason for handing in an assignment late ... so make sure to submit the assignment WELL BEFORE it is due !

- You WILL lose marks on this assignment if any of your files are missing. So, make sure that you hand in the correct files and version of your assignment. You will also lose marks if your code is not **written neatly with proper indentation and containing a reasonable number of comments.** See course notes for examples of what is proper indentation, writing style and reasonable commenting).