Problem Set 4: Structures and Sorting

Please send back via NYU Brightspace

- A zip archive named as
  `PS04_Last_First.zip`
  Where `First` and `Last` are your first and last name.
  And the zip archive contains your **`sort_struct.c`** file.

Look up how to use any of the functions you might want to use at https://www.cplusplus.com/. Look at example code to determine what `#include <>'s` are needed in addition to `#include <stdio.h>`.

Total points: 100

Problem 1
Sort an Array of Structures
100 points

Points are awarded as follows:

- **25 Points** – read in struct Student data from input file
- **30 Points** – sort data using qsort() (10 points for each sort-on key)
- **25 Points** – write out struct Student data to output file
- **20 Points** - clear code, sensible formatting, good comments

You are given the files **sort_main.c and sort.h**. These files are complete and do require any additional code. It is your task to create **sort_struct.c** that contains all of the functions indicated in **sort.h**, namely:

```
read_students()
sort_students()
comp_first()
comp_last()
comp_id()
write_students()
```

You can use the bash script **`./build.sh`** as an easy way to compile your program.

Description of sort_main.c
File **sort_main.c** takes 3 command line arguments, in the following order:

Sort-on field name as the literal: `first` or `last` or `id`
`ifile` (which must be `name_id.csv`)
`ofile` (which can be any file named as `*.csv`)

If the command line arguments are incomplete, it prints a usage statement and returns -1, otherwise it parses the command line arguments.

It opens `ifile` for reading and `ofile` for writing. In the file `name_id.csv`, each line has three fields separated by a comma with the line terminated by '\n':

```
lastName,firstName,idNumber
```

where `firstName` and `lastName` are alpha characters and `idNumber` is numeric characters.

It reads all lines in `ifile` and use each line to fill the array `struct Student sdata[]`. It prints the student data to the terminal and prints the number of students that were read in.

It sorts the array `struct Student sdata[]` according to the sort-on key `id`.

It writes the sorted data to `ofile`.

It closes the file pointers and returns.

Description of functions you must write

These functions will be file **`sort_struct.c.`** The function prototypes for each function are in **`sort.h,`** which must be an `#include` at the top of your file:

```
#include "sort.h"
```

`int read_students(FILE *ifp, struct Student *sdata)`

This function's arguments are the input file pointer and a pointer to `sdata`. It returns the number of students read.

Remember that since `sdata` is an array of structures, in this function you can use `sdata[i].last` (or `first, id`) to access the elements of the structure at index `i`.

For each line in the input file, your function should:

- use **`fgets()`** to read a line from the file.
- use **`strtok`**() to parse the line into `first, last` and `id`. Since `fgets()` includes the `'\n'` in each line of the file, use `",\n"` (comma and newline) as the delimiter string in `strtok`().
- Copy the `first` and `last` name strings from `strtok()`into the corresponding elements of the struct at index i, and write the id value into the corresponding elements of the struct at index i.

After all lines have been read

- Return the number of students read

Note that in **`struct_main.c`**, `sdata[]` is declared as

```
struct Student sdata[MAX_STUDENTS];
```

so it is an error to read more than `MAX_STUDENTS` lines from the input file and write the results into sdata[]. You could use a `for()` statement with upper limit of `MAX_STUDENTS` to enforce this limit. In general, there will be fewer than `MAX_STUDENTS` lines in `ifile`, so break out of the `for()` loop when `fgets()` returns `NULL`, i.e. when there are no more lines to read.

Look up how to use `fgets()` and `strtok()` at https://www.cplusplus.com/. Look at example code to determine what `#include <>'s` are needed in addition to `#include <stdio.h>`.

`int sort_students(char *sort_key, int num_students, struct Student *sdata)`

This function's arguments are a pointer to the sort_key and the number of students, in sdata[]. This function in turn calls

```
qsort(&sdata[0], num_students, sizeof(sdata[0]), comp_ftn_ptr);
```

where ***`comp_ftn_ptr`*** is a pointer to one of the following functions:

```
comp_first(const void * a, const void * b);
comp_last(const void * a, const void * b);
comp_id(const void * a, const void * b);
```

Remember that a pointer to a function is simply the name of the function without the parenthesis and arguments. So `comp_first` is a pointer to the function

```
comp_first(const void * a, const void * b);
```

The first lines of comp_first() must cast its arguments as pointer to struct:

```
struct student *pa = (struct student *)a;
struct student *pb = (struct student *)b;
```

You finish up comp_first() with this line:

```
return(strcmp(pa->last, pb->last));
```

Which is all you need to do, since strcmp() returns -1 if the first string is alphabetically before the second, 0 if they are the same and +1 if first string is alphabetically after the second.

Look up how to use `strcmp()` and `qsort()` at https://www.cplusplus.com/. Look at example code to determine what `#include <>'s` are needed in addition to `#include <stdio.h>`.

`void write_students(FILE *ofp, int num_students, struct Student *sdata);`

This function's arguments are the output file pointer, the number of students in `sdata[]` and a pointer to `sdata`. It has no return value, i.e. its return value is `void`.

For every student in `sdata[]`, use `fprintf()` to write `last, first` and `id` fields to ofile in CSV format using format string:

```
"%s,%s,%d\n"
```

Also print the line to the terminal.