CIS*2500 (Intermediate Programming) Lab #1

Due date: Week 2 Friday January 22nd at 11:59PM EST (**page 2 has** instructions on submission)

Concepts: structures, array of structures, functions and text files

Description: For this lab, assume the following definitions:

```
#define SIZE 25
#define NUM_EMP 3

/* definition of an employee record*/
typedef struct employee
{
   char fname[20];
   char lname[20];
   int id;
   char dependents [3][20]; // assume that no employees can have more
                            // than 3 registered dependents
} Employees;
```

You are required to write function definitions for the following tasks and a main to test them.

1. Print information of *c* number of employees. This function takes 2 parameters - an array of type Employees and an integer variable (e.g. *c*).

Function name: printEmployees
Prototype: void printEmployees (Employees [NUM_EMP], int);

2. Save information of *c* number of employees in a text file. This function takes 3 parameters - an array of type Employees, an integer variable (e.g. *c*) and a string that holds the name of the file. Note that this name must be accepted from command line and then passed to this function.

Function name: saveEmployees
Prototype: void saveEmployees (Employees [NUM_EMP], int c, char [SIZE]) ;

3. Load all employee records stored in a file to an array of Employees. It returns the total number of records loaded.

Function name: loadEmployees
Prototype: int loadEmployees (Employees [NUM_EMP], char [SIZE]) ;

4. Swap information of 2 Employees.

Function name: swapEmployees
Prototype: void swapEmployees (Employees *, Employee *);

Submission Instructions:

1. Create 2 C files for this lab – one that has the function definitions and the other that has the main program to test the functions. You may or may not create and use a separate header file for this lab.
2. Create a make file that compiles your c files and creates an executable. For example, if my c files are called lab1.c and lab1Main.c, and makefile has the following content, then running **make** utility will create an executable file called lab1_output. It can then be run with a command-line argument that holds the filename – for example,
./lab1_output fileEmployees

----------start of makefile----------
```
lab1_output: lab1.o lab1Main.o
	gcc lab1.o lab1Main.o -o lab1_output

lab1.o: lab1.c lab1.h
	gcc -Wall -std=c99 -c lab1.c

lab1Main.o: lab1Main.c lab1.h
	gcc -Wall -std=c99 -c lab1Main.c

clean:
	rm *.o lab1_output
```
----------end of makefile----------

3. Submit all your files to Gitlab.
- makefile
- lab1.c
- lab1Main.c
- lab1.h (if you are using a header file)

Follow these instructions to use gitlab:

Step 1: Computer setup (local)
- make sure git is installed (https://git-scm.com/downloads)
- Mac users can use the terminal mode (I have tested it on my mac)
- Windows users can use powershell, WSL (windows subsystem for linux) or git bash.
- decide where cis2500 work will go and make a directory (e.g. mkdir CIS2500)
- cd to that directory from the terminal application

Step 2: From the chosen directory, now type:

git clone https://git.socs.uoguelph.ca/2500w21/<your username>/lab1.git

At this point, you have a directory to work with on your local system.

Step 3: Do the work

- (remember to) cd to the new directory
- create the file that you are working on
- after first save, type: git add filename

ONLY ADD THE FILE ONCE!!! //do not do: git add .

Loop every 20-30 minutes:

```
git commit -am "write something here about what you just did"
```

Once per day:

```
git push   // this is what stores local work back to the server.
```

To learn more about Gitlab, go to this link on moodle
https://moodle.socs.uoguelph.ca/course/view.php?id=169