Lab7
Deadline: In lab in the week of Mar 25

Requirements
We are writing a program that stores information of the students in a class and enables user to manipulate the data regarding each student. The program has the following features:

Each student in class has these attributes:
Last name
First name
Student number
Midterm Grade
Final Grade

Prints the list of students in alphabetical order of their last names. Depending on which option is selected, the grades of all students are written to the output file based on the following rule:
1: For an average higher than 90% (inclusive)
2: For an average between 80% (inclusive) - 90% (exclusive)
3: For an average between 70% (inclusive) - 80% (exclusive)
4: For an average between 60% (inclusive) - 70% (exclusive)
5: For an average below 60% (exclusive)

Here is the sample input:
```
>>cat input.txt
1
Jane Rogers A0000001 100 100
Kim Hazelwood A0000003 95 90
John Kim A0000004 0 90
Peter Rodriguez A0000005 50 62
E
```

Here is the output based on the input above:
```
>>cat output.txt
Jane Rogers A0000001 100 100
Kim Hazelwood A0000003 95 90
```

Implementation Details
1. The first number indicates the option number.
2. Student number is in the same format as your A number. Let's assume that it is A followed by 7 digits.
3. The average is calculated by (midterm grade + final grade)/2.
4. After the filtering based on the option is done, the sorting criteria is ascending order (meaning smallest to largest) last name > first name > student number > midterm grade > final grade.

Up until now, this lab sounds very manageable. However, this lab has a special component where you are trying to protect your program as much as possible. The user can try very strange things such as putting a negative number. What you are supposed to handle are:

1. The midterm and the final grade are any real number ranges from 0 to 100 inclusive.
2. Everyone has a first and last name
3. Everyone has an A number

Your code must be able to output "Error" text to the output file upon detecting something that is not considered a normal operation. The output file is guaranteed to be valid, so no error checking needs to be done on the output file.

How to Compile and Run

- The Makefile for lab is provided.
- The Makefile is supposed to work with lab6.c, input.txt, output.txt and ref.txt files so, make sure to name your files accordingly.
- Run the following command in vs code Terminal.
  `make`
  It should compile the code without any errors.
  `make convert_input`
  It should convert the input.txt file to unix encoding.
  `make run`
  It should run the compiled code.
- Run the following command to delete the out file.
  `make clean`
- Run the following command to convert the generated output to unix encoding.
  `make convert_output`
  It should convert the output.txt file to unix encoding.
- Run the following command to check your output with provided ref file.
  `make check`
- You are not supposed to make any changes in the Makefile.
- Make sure to install dos2unix utility using the following command:
  `sudo apt-get install dos2unix`
  For Mac
  `brew install dos2unix`

Grading

Any grading failure due to not following instructions will result in 0. You will get one chance to show your work to the instructor.

- (1 point) All files are submitted correctly using the instructions.
- (1 point) Generate a correct solution to the problem(s) in this lab.

- (2 point) Proper error handling.

Submission Files

- You must submit only one file named to Learning Hub: **lab7.c**
- Submit it to learning hub before the lab session