4.1 UCupiD

A group of sociology students have had the idea of creating a dating application for UC Davis, and even found a name: **UCupiD**. However, none of them know how to program and they need help implementing their idea.

A file database has an entry for each individual who has subscribed to the service. Each entry contains the individual's SID number (9 digits), first name (< 20 characters), sex (M or F), sexual orientation (M or F), age, political leaning (from 0 for very conservative, up to 10 for very liberal), major hobby (< 20 characters), minor hobby (< 20 characters), height (in inches) and weight (in pounds). Entries are ordered by SID numbers.

Two individuals match if their respective sex and sexual orientation are compatible (e.g. a female looking for males would be compatible with a male looking for females, or a male looking for males would be compatible another male looking for males, etc.) and if one of the following matching criteria is triggered:

1. *Social identity*: the two individuals are within 10 years of each other and have a political leaning within 2 points of each other.

2. *Personality*: the two individuals share both their major and minor hobbies.

3. *Appearance*: the two individuals are within 10% of each other's height and weight.

Write program `UCupidD.c` which receives as program arguments the name of the database file and an individual's SID, and generates an output file containing the list of possible matches for this individual, ordered by their SID, as illustrated in the following example.

```
$ cat database.txt
111223333,Rob,M,F,47,5,Dancing,Painting,63,165
123456789,Bob,M,F,77,2,Baseball,Cooking,72,275
194783218,Ann,F,M,28,9,Dancing,Painting,65,125
222334444,Barbara,F,M,40,4,Jogging,Movies,64,145
222334567,John,M,M,45,7,Cooking,Hiking,70,182
555446767,Joan,F,M,69,1,Organizing,Writing,59,115
622555555,Drewbert,M,M,59,6,Organizing,Fishing,68,170
823810924,Andrew,M,F,25,4,Dancing,Painting,84,280
622555556,Zeta,F,F,34,6,Organizing,Fishing,68,170
823810925,Fran,F,F,25,10,Dancing,Painting,84,280
847111411,Adam,M,M,82,7,Theatre,Sparring,72,200
999999999,Aerith,F,M,22,1,Gardening,Cooking,63,100
$ ./UCupiD database.txt 111223333
$ cat match_111223333.txt
Matches for user 111223333 (Rob):
- 194783218 (Ann)
- 222334444 (Barbara)
$ ./UCupiD database.txt 622555556
$ cat match_622555556.txt
Matches for user 622555556 (Zeta):
- 823810925 (Fran)
$ ./UCupiD database.txt 000000000
User with SID 000000000 not found
$ ./UCupiD wrong_file.txt 000000000
Error: cannot open wrong_file.txt
```

```
$ ./UCupiD
Usage: ./UCupiD db_file SID
$ echo $?
1
$
```

Here are a list of requirements, assumptions and hints:

- The program should receive two command-line arguments, the name of the database file and an individual's SID. If less than two arguments are provided, the program should display an error message and return with error code `1`.
  - Other errors include: database file cannot be opened, individual cannot be found in database, output file cannot be created and opened.
- The name of the output file must be generated based on the individual's SID for which the program is trying to find matches: `match_<SID>.txt`.
- The database file is formatted in `csv`, which means that each individual's entry is described on one line, and the information within each line are separated by commas. The format for an entry is:
- ```
  SID,Firstname,Sex,Orientation,Age,PoliticalLeaning,MajorHobby,
  MinorHobby,Height,Weight
  ```

  For reading the file, it is recommended to use `fgets()` to read an entire line (we assume that the maximum amount of characters per line is 256) and `sscanf()` to parse the entry and retrieve each of the individual's information.

  Because commas are not considered as a word delimiter by `sscanf()` when using `%s`, we need to use string specifier `%[^,]`. This specifier will match all the characters until encountering a comma. Here an example code:

  ```
  char *entry = "Homer,45";
  char name[20];
  sscanf(entry, "%[^,]", name);
  printf("%s", name);              /* Will print 'Homer' */
  ```

- When reading the database file, you have to create a linked list to represent individuals in your program.

  IMPORTANT: Not using a linked list will result in getting a straight zero for the *Implementation* score on Gradescope.

- Here is the suggested order of implementation for this program:

1. Define a structure that represents each individual.
2. Write a function that reads each entry from the database file, and creates a structure object that is inserted in a linked list of individuals.
3. Write a function that finds an individual in the linked list based on their SID.
4. Write a function that finds all the matches between an individual and all of the individuals represented in the linked list, and outputs the matches in a file.
5. Write a function that destroy the linked list.
6. Write the `main()` function which receives the command-line arguments and then calls all the functions above in the appropriate order.

- List of some important libc functions that are used in the reference
  ```

program: `ropen()`, `rclose()`, `rgets()`, `sscanf()`, `malloc()`, `free()`, `strcmp()`, `sprintf()`, `abs()`.