COMP 3958: Lab 4

Submit a source file named `records.ml`. You may use functions in the standard modules (except the `Str` module which deals with regular expressions). You are not allowed to use `for` or `while` loops. Your file must build without warnings or errors. Otherwise, you may not receive credit for this lab. Maximum score: 15

Write a stand-alone Ocaml program to sort records read from a file whose name is specified on the command-line. If no file is specified on the command-line, the program prints the expected usage and exits. If more than one file is specified, only the first file is processed and the rest are simply ignored. If the specified file cannot be opened, print an error message and exit the program.

Each input line contains information of at most one record. The following shows 5 sample input lines:

```
homer  simpson  25  # bad at nuclear engineering
 ned flander  abc  !! invalid
   waylon   smithers  100  ! high score
  gary  chalmers  5
monty  burns
```

For a line to be a valid record, it must have at least 3 words and the third word must be an integer between 0 and 100 inclusive. (Words are separated by one or more space characters. Note that the second and the last input lines above are invalid.) The first word is regarded as the first name, the second word as the last name, and the third word as the score. Any extra text after those 3 words is regarded as a comment. The program needs to make sure that a line it reads has at least 3 words and that the third word is an integer between 0 and 100 inclusive. If not, that line is invalid and is skipped. Note that there are no restrictions on the first name and the last name except that each is a word.

Use the following type to store a record:

```
type record = {firstname: string; lastname: string; score: int}
```

After reading all valid records, the program proceeds to sort them. The sorted output should be in descending order of scores, and if multiple records have the same score, they are then in ascending order of last names. Finally, records with the same score and last name are in ascending order of their first names. The following shows the output format for the above input:

```
100 smthers waylon
 25 simpson homer
  5 chalmers gary
```

Note that the output starts with the score followed by the last name, and then by the first name. Comments are not displayed in the output. Sorted records are displayed to standard output.

Name your source file `records.ml`. We'll be building your program using the command: `ocamlbuild records.native`. Make sure your program can be successfully built with this command.

To facilitate testing (in case your program does not quite work), there must be the following two functions:

```
val sort_records : record list -> record list (* sort list of records in specified order *)
val parse : string -> record option  (* get a valid record from a line if possible *)
```

Be sure to provide comments to indicate what each function does.