AC21007 Assignment 5.
Common Lisp

The goal of this assignment is to implement a small project in Common Lisp.

Description of the problem. In this assignment, you will write a Common Lisp database to keep track of mp3 songs. The database will make it possible to add, delete, select and update songs. Our database will have the following fields:

- **id**: a natural number to identify the song.
- **song**: the name of the song (a string).
- **group**: the name of the group that plays the song (a string).
- **year**: the year when the song was released (a natural number higher than 1900).
- **length**: the length of the song in seconds (a natural number).
- **rate**: the rate of the song (one of the rational numbers: 0/5, 1/5, 2/5, 3/5, 4/5 or 5/5).

An example of a database with these fields is given as follows:

id	song	group	year	length	rate
1	Somebody to love	Queen	1976	296	5/5
2	A day in the life	The Beatles	1967	335	4/5
3	Hey Brother	Avicii	2013	254	3/5

The representation of the above database in Lisp is given by the following list of lists:

```
'((1 "Somebody to love" "Queen" 1976 296 5/5)
  (2 "A day in the life" "The Beatles" 1967 335 4/5)
  (3 "Hey Brother" "Avicii" 2013 254 3/5))
```

Basic functionality. You need to implement the following functionality:
F1. A predicate function `songp` to recognise if a song is well-formed. A song is well-formed if it is a list with 6 elements and those elements satisfy the specification given for the fields of our database. Example:

```
> (songp '(1 "Somebody to love" "Queen" 1976 296 5/5))

t
```

F2. A predicate function `mp3databasep` to recognise mp3 databases. A mp3 database is a lists of songs. Example:

```
> (mp3databasep '((1 "Somebody to love" "Queen" 1976 296 5/5)
                  (2 "A day in the life" "The Beatles" 1967 335 4/5)
                  (3 "Hey Brother" "Avicii" 2013 254 3/5)))

t
```

F3. A function `insertsong` that inserts a song `s` in an mp3 database `d`. Before inserting the song, the function should check that `s` is a song, `d` is an mp3 database, and the id of `song` is different to the ids in the database. Example:

```
> (insertsong '(4 "I want to break free" "Queen" 1984 200 5/5)
              '((1 "Somebody to love" "Queen" 1976 296 5/5)
                (2 "A day in the life" "The Beatles" 1967 335 4/5)
                (3 "Hey Brother" "Avicii" 2013 254 3/5)))

((1 "Somebody to love" "Queen" 1976 296 5/5)
 (2 "A day in the life" "The Beatles" 1967 335 4/5)
 (3 "Hey Brother" "Avicii" 2013 254 3/5)
 (4 "I want to break free" "Queen" 1984 200 5/5))
```

F4. A function `removesong` that removes the song with id `n` from an mp3 database `d`. Example:

```
> (removesong 3 '((1 "Somebody to love" "Queen" 1976 296 5/5)
                  (2 "A day in the life" "The Beatles" 1967 335 4/5)
                  (3 "Hey Brother" "Avicii" 2013 254 3/5)
                  (4 "I want to break free" "Queen" 1984 200 5/5)))

((1 "Somebody to love" "Queen" 1976 296 5/5)
 (2 "A day in the life" "The Beatles" 1967 335 4/5)
 (4 "I want to break free" "Queen" 1984 200 5/5))
```

F5. A function `updaterate` that updates the rate of the song with id `n` from an mp3 database `d`. Example:

```
> (updaterate 3 2/5 '((1 "Somebody to love" "Queen" 1976 296 5/5)
                      (2 "A day in the life" "The Beatles" 1967 335 4/5)
                      (3 "Hey Brother" "Avicii" 2013 254 3/5)
                      (4 "I want to break free" "Queen" 1984 200 5/5)))

((1 "Somebody to love" "Queen" 1976 296 5/5)
 (2 "A day in the life" "The Beatles" 1967 335 4/5)
 (3 "Hey Brother" "Avicii" 2013 254 2/5)
 (4 "I want to break free" "Queen" 1984 200 5/5))
```

F6. Four functions to sort the songs of a database by group, year, length and rate respectively. Example:

```
> (sortyear '((1 "Somebody to love" "Queen" 1976 296 5/5)
              (2 "A day in the life" "The Beatles" 1967 335 4/5)
              (3 "Hey Brother" "Avicii" 2013 254 3/5)
              (4 "I want to break free" "Queen" 1984 200 5/5)))

((2 "A day in the life" "The Beatles" 1967 335 4/5)
 (1 "Somebody to love" "Queen" 1976 296 5/5)
 (4 "I want to break free" "Queen" 1984 200 5/5)
 (3 "Hey Brother" "Avicii" 2013 254 2/5))
```

F7. A function `longestsong` that returns the longest song.

```
> (longestson '((1 "Somebody to love" "Queen" 1976 296 5/5)
                (2 "A day in the life" "The Beatles" 1967 335 4/5)
                (3 "Hey Brother" "Avicii" 2013 254 3/5)
                (4 "I want to break free" "Queen" 1984 200 5/5)))

(2 "A day in the life" "The Beatles" 1967 335 4/5)
```

[Maximal mark – 55]

Extensions. There are several extensions that you could pursue.
E1. Use different sorting algorithms for the functions of **F6**. The sorting algorithm can be given as an argument of those functions.
E2. Define a function to search all the songs of a given group.
E3. Define a function to search all the songs that were released between two given dates.
E4. Define a function to search all the songs whose title contain a given word.
E5. Define a function to return the songs with the maximum rate.
E6. Define functions to access the different components of a song.
E7. Download from MyDundee the files `database.txt` and `database-access.lisp`. The `database-access.lisp` contains two functions `read-database` and `write-database` to read from the database `database.txt` and write in the database `database.txt`, respectively. Include the file `database-access.lisp` in your development and use the functions `read-database` and `write-database` in the functions that you created for **F3**–**F7**.
E8. Define a function to return a random song (you can check the use of the random function in `http://www.ai.mit.edu/projects/iiip/doc/CommonLISP/HyperSpec/Body/fun_random.html`).
You can define other functions that you consider useful.

[Maximal mark – 25]

Top extensions. The top extensions require the study of some Common Lisp advance features that are not explained in the module (and therefore will not be included in the exam).
T1. Pretty print the output of your functions using the `format` function (see `http://www.ai.mit.edu/projects/iiip/doc/CommonLISP/HyperSpec/Body/sec_22-3.html`).
T2. As we have seen in the lectures, there are some functions that can take an arbitrary number of arguments (for example, `+` or `append`). This functionality is

achieved including the `&rest` keyword in the definition of your functions. This functionality is explained in Chapter 20 of Shapiro's book. You could use this feature to, for instance, search all the songs that were released in an arbitrary number of years.

T3. Some Common Lisp functions can have optional arguments (for example, the `subseq` function). This functionality is achieved including the `&optional` keyword in the definition of your functions. An explanation of this feature is given in pages 186 and 187 of Shapiro's book. You could use this functionality in the definition of the function to search all the songs that were released between two given dates (**E2**). The second argument of the function could be declared as optional; then, if only one year is given as input, the function will return the songs released from that year up to now.

T4. Some Common Lisp functions have key parameters (see the `:count` keyword parameter of the `remove` function). This is obtained using the `&key` keyword as explained in pages 223 and 224 of Shapiro's book. You could use this feature in the functions to sort the songs including a keyword argument to select the sorting algorithm to use.

[Maximal mark – 20]

Submission & Assessment. This assignment is due to for submission via MyDundee (under AC21007 Assignments) on 3 April and is worth 10% of your total grade for this module. Keep a copy of what you submit in case there are problems with your submission. As well as the source code for your lisp database, you should include a report of no more than one page that explains how you approached the assignment, the problems that you faced, and the solutions that you have created.

Your submission should be a ZIP file containing:

- your Lisp file(s) and;

- your report (as a PDF document).

Marking scheme. The marks for the assignment will be calculated based on the following:

1. Functionality implemented.
2. Quality of the code. The use of comments, documentation of functions, use of assertions to manage errors will improve the quality of your code.
3. Report quality. Good style, sections including motivation for the problem, description of any technical hurdles encountered, indication that alternative solutions/algorithms were considered, comparison with examples from lectures and textbooks, and good logical conclusion – are all signs of a good report.