Task 1: Reordering Footnotes

A text contains footnotes in square brackes, holding numerical reference numbers. At the end of the text, in a special section, all targets of the references are listed. Each footnote points to exactly one target entry, but there may be more than a single reference to each target. The ordering of references and targets is unordered. Write a program that reorders the target footnotes beginning with 1 and subsequently incrementing. Optionally user should be able to choose to order the footnotes by their first appearance in the text.

Format

The text is in plain ISO-8859-15 encoding containing brackets like

```
The fox jumped [13]
```

The references may appear anywhere in the text, but will be completely on one line. More than one single reference can appear on a single line.

At the end of the text there is a footnote section starting with a marker

```
@footnotes:
```

on a line of its own. After this marker no more references occur. Every line that starts with square brackets is a target:

```
[13] John Doe and Ed Marten: "About Foxes", 1873, University Press
[9] Linus Torvalds: "Linux Kernel", 1991 - 2008, published in the Internet
```

Specification

The program takes a single text from a command line argument containing a filename and produces output on stdout. The texts can be of varing length from a couple kBytes up to several hundred MBytes.

Sample data

```
A great brown fox [13] jumped of a pile of lorem ipsum [4], [7]. He met
with a silver penguin, browsing the Linux Kernel Mailinglist [3]. They
debated other the question whether to start a C-program with "main
(int argc, char **argv)" or with "main (int argc, char *argv[])".
Square brackets annoyed them [9999].

@footnote:

[13] Al Fabetus: "On characters and animals", 1888, self published.
[4] Lorem Ipsum, <a href="http://en.wikipedia.org/wiki/Lorem_ipsum">Web Link</a>
[9999] Annoying Link.
[7] B. Fox: "More on Blind Text".
[3] Linux Kernel Maintainers: LKML
```

Task 2: Simple Music Management

Write a web based application that is able to maintain a personal music collection that is stored in a database.

Data model

A data record of the table "music" consists of just four attributes:

1. Title of the song (string up to 72 characters),
2. Artist (string up to 72 characters),
3. Publishing year (a four digit interger number), and
4. Genre (a numerical foreign key to the "id" attribute of the table "genres").

A database schema is provided that implements the tables "music" and "genres". A database dump contains SQL code that populates the database with some sample records. It is not allowed to use any database features like triggers, stored procedures, autoincrements, or the like.

Application requirements

The application should be able to perform the following tasks:

- Enter a new database record
- Search for a database record
- Edit an existing database record

Deleting records is not necessary. Duplicate records or duplicate attributes are allowed. To search for a record, the user may provide values to one or more attributes. The program should display all records that match exactly the entered attributes.

Allowed prerequisites

You may use a database of your own choice that is available via normal package management of a common distribution like MySQL or PostgreSQL. You may use extra libraries, modules, or packages for your programming language, but you need to explain how to install them via normal package management of a common

distribution.

Database definition

```
CREATE DATABASE jukebox;
```

```
CREATE TABLE music (
    title VARCHAR(72),
    artist VARCHAR(72),
    year INT NOT NULL,
    genre INT NOT NULL
);
```

```
CREATE TABLE genres (
    id INT,
    desc VARCHAR(20)
);
```

Sample Data

```
INSERT INTO genres VALUES (1, 'Rock');
INSERT INTO genres VALUES (2, 'Pop');
INSERT INTO genres VALUES (3, 'Classical');
INSERT INTO genres VALUES (4, 'Folk');
INSERT INTO genres VALUES (5, 'Jazz');
```

```
INSERT INTO music VALUES ('Money for Nothing', 'Dire Straits', 1984, 2);
INSERT INTO music VALUES ('Hey, Joe',          'Jimi Hendrix', 1966, 1);
INSERT INTO music VALUES ('Like a Virgin',     'Madonna', 1984, 2);
INSERT INTO music VALUES ('Yesterday',         'Beatles', 1965, 2);
INSERT INTO music VALUES ('Country Roads',     'John Denver', 1971, 4);
INSERT INTO music VALUES ('Toccata and Fugue', 'Johan Sebastian Bach', 1703, 3);
```