SEng 265 — Introduction to Software Engineering
Fall 2002
Assignment No. 1

Note 1 **This assignment is to be done in teams of two people.**

Note 2 Except as indicated in the classroom, working with other teams is strictly prohibited.

Note 2 Your assignment will be tested on aserver.

- Due date: Sept. 25, 2002, at the beginning of the class.
- This assignment is worth 20 points.
- This assignment is worth 5% of your total course mark.
- Clearly mark your lab section and student number on all submissions.

Objectives

After completing this assignment, you will have:

- learned the basics of Unix pipes, and how to use simple commands to accomplish complex tasks.

Introduction

I have a large library of mp3s on one of my linux boxes. I keep track of what songs I have in a text file with the following format:

Field Number	Columns	Description	Comments
1	1-30	Band name	
2	32-35	Year of album	Always 4 digits
3	37-66	Title of album	
4	68-69	Track number	Always 2 digits
5	71-110	Song title	

The field separator is always the **:** (colon). You can assume that this character only appears as separator and never as part of the data. The following chunk of the file represents an example of how an album is represented (the first line is an aid that indicates the column numbers and it is not part of the file). It corresponds to the album *Animals* by *Pink Floyd*, recorded in *1977*, which contains 5 tracks.

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901
Pink Floyd                :1977:Animals                     :02:Dogs                                    :
Pink Floyd                :1977:Animals                     :01:Pigs on the Wing Part 1                 :
Pink Floyd                :1977:Animals                     :04:Sheep                                   :
Pink Floyd                :1977:Animals                     :03:Pigs Three Different Ones               :
Pink Floyd                :1977:Animals                     :05:Pigs on the Wing Part 2                 :
```

Please note that the records might not be sorted.
My collection of music is getting to the point where it is difficult to know what I have. I need a collection of programs that help me analyze it.

Your task, should you choose to accept it

I want you to write ten different bash shell scripts, each of which does the following:

1. Print the number of songs in the file.
2. Print the number of albums in the file.
3. Print the number of songs by U2. (Hint, try using `"^U2"` as a parameter to `grep`.)
4. List, in cronological order, the title of the albums by U2.
5. List only the song titles of the first song of each album by U2. The resulting list should be in reverse alphabetical order and should not include duplicates. You can safely assume that each album will start with track number 01.
6. Print the name of the album that has the biggest number of tracks. You can assume that there is only one album with this number of tracks.
7. Print the number of tracks in the album with the biggest number of tracks.
8. List only the names of the albums, in chronological order. (Albums from the same year should be ordered in alphabetical order)
9. Order the records by song title (songs with the same title should be ordered in the year in which they appeared), then print the entire record of the penultimate record in the resulting order.
10. Print, in alphabetical order, the title of those songs which exist in more than one album.

Implementation notes

- The name of each script should be **jukebox_yy**, where *yy* corresponds to the task number (00, 01, 02, ..., 10, e.g. jukebox_01).
- Each script should be called with one parameter: the name of the file to scan. You can safely assume that the script will always be called this way.
- Your scripts could only include *bash* shell scripting, and calls to typical Unix commands (as found in **aserver**, where the scripts are expected to run). These commands should be combined by using pipes or by saving intermediary results in temporal files. All the tasks can be accomplished in less than 10 lines, and without any programming involved.
- Any temporal file should be created in the current directory.
- Your program can safely assume that all the input conforms to the expected format.
- Any unresolved issue with respect to this assignment will be discussed in the bulletin board.
- You can find an example of an mp3 songs file from `/home/seng265/assign1/music.txt`. Your program will not necessarily be tested using this file.
- You can also find the expected output of each command in the files `/home/seng265/assign1/example_results_yy`, where yy corresponds to the test number. These results were computed using `music.txt` as input. Your scripts should generated **exactly** the same output when run on `music.txt`. Use `diff` to verify this.

Additional Hints

- Check the man pages for the following commands: `sort`, `head`, `tail`, `cat`, `wc`, `grep` and `diff`.
- If you need to replace spaces and tabs from the output of a command (e.g. if you want to use its output as a parameter to another command) you can use the following command:

```
sed 's/[\t ]//g'
```

What to submit

- Submit your source code (the ten files `jukebox_yy`) via CVS, in a directory called `assign1` in the module of each of the members of the team (yes, that means that both members should do the check-in the files).
- At the start of the class on the deadline, **hand in a printed copy of your `cvs commit` script as well as a hard copy of all your scripts**.