CW-B2 Audioscrobbler Artists

Released: Nov 18th 2016
Deadline: Dec 6th 2016
Weight: 15 %

This coursework carries 15 % towards your final mark. Once you have completed the two tasks, submit one C source file `topk.c`. by clicking the "Submit CW-B2" link on Moodle. You can ask a lab assistant to give you a mark and feedback during the lab hours, or in one of the coursework marking sessions. The marking scheme is as follows:

- ▷ Task 1: 1 mark for a), c), d), and e). 2 marks for b).
- ▷ Task 2: 2 marks for a), b), and d). 3 marks for c).

Introduction

Audioscrobbler was an online service for storing music listening patterns. A data set was published to the public domain containing a list of artists and user plays.

In the previous coursework, we created functions for manipulating the play data set. In this coursework, we will also use the artist data given in the file `artist_data.txt`. Here are the first three lines of this file:

```
1134999 06Crazy Life
6821360 Pang Nakarin
10113088 Terfel, Bartoli- Mozart: Don
```

Each line consists of two columns, an artist id and an artist name. The two columns are separated by a tab character `'\t'`.

Task 1 - Representing Artists

In this task, we will create functions for manipulating linked lists of artists. You can solve this task by reusing ideas from the solution to coursework CW-B1.

a) Below, we have started defining a struct `artist` in which we will store artist data. Complete the definition such that it can be used as a linked list, where each element in the linked list contains the artist id, the artist name, a play count, and an alternative artist id represented by an `int` if the artist is misspelled.

```
struct artist { ... };
```

b) Implement a function with the following type declaration which reads the artist data from the file with the given file name into a linked list of artists. The function should return a pointer to the head of the linked list, or return `NULL` and print an appropriate error message if there was a memory allocation error or a file error.

```
struct artist *read_artists(char *fname);
```

Hint: Some input lines are long. You can ignore artists whose artist names are longer than 64 characters. You can use the following format string when parsing a line of the artist data file.

```
"%d\t%65[^\t\n]\n"
```

c) Create a function with the following type declaration

```
void print_artist(struct artist *a);
```

which outputs the artist id, the artist name, and the play count for an artist. The output should be on the format below.

```
artist name (artist id) [playcount]
```

If `p` is a `NULL` pointer, then the function should instead print `NULL`.

d) Construct a function which prints all artists in a linked list of artist. Each artist should be printed on the form given by sub-task c)

e) Define a function with the following type declaration which deallocates (frees) the memory used used by a linked list of artists.

```
void free_artists(struct artist *a);
```

Task 2 - Finding Top Artists

The purpose of this task is to find the artists which are played the most. You need to re-use parts of the solution to coursework CW-B1 to solve this task.

a) Implement a function `sort_plays` which sorts a list of plays according to increasing artist id. The function should take as argument a pointer to a linked list of plays, and should return a pointer to a sorted linked list of plays. Hint: Use the merge sort algorithm.

```
struct play *sort_plays(struct play *head);
```

b) Implement a function `sort_artists` which sorts a list of artists according to increasing artist id or decreasing play count. You should use the following macro-defined constants in the top of your C code to represent the sorting criteria.

```
#define PLAYCOUNT 0
#define ARTISTID  1
```

The function should be declared as follows.

```
struct artist *sort_artists(struct play *artist, int criterion);
```

The second argument is the sorting criterion, and should be either `PLAYCOUNT` or `ARTISTID`. The function should print an error message and return the list unsorted if any other sorting criterion is given.

Hint: Do not start solving this sub-task until you have solved sub-task a) correctly.

c) The file `artist_data.txt` contains only the artist ids and the artist names, and not the number of times the artist has been played, i.e., the play count.

We will now update the play counts using the play data set. Recall that each line in the play data set corresponds to the number of times a user has played an artist. The play count of an artist should be the sum of all times the artist has been played by the users.

Create a function which given a list of artists and a list of plays, updates the play count field for all artists based on the play data. The function should return a linked list of the artists.

Hint: Sort the two lists according to artist id before updating the play counts.

```
struct artist *update_counts(struct artist *a, struct play *p);
```

d) The final task in this coursework is to integrate all the functionality we have implemented into a program `topk` which prints a list of the k most played artists. The program should take the following three arguments:

1) k, i.e., the number of artists to play
2) the name of the artist data file, and
3) the name of the play data file.

If any error occurs, then the program should print an appropriate error message and halt.