CS302 -- Lab 1

- CS302 -- Data Structures and Algorithms II
- James S. Plank
- This file: **http://web.eecs.utk.edu/~jplank/plank/classes/cs302/Labs/Lab1**
- Lab Directory: **/home/jplank/cs302/Labs/Lab1**

What you are to submit

You are to submit the file `lib_info.cpp` on Canvas. Do not submit anything else. That should contain all of your code, and should compile with no errors or warnings when you do:

```
g++ -Wall -Wextra -std=c++98 -o lib_info lib_info.cpp
```

---

lib_info - A program to help you parse your music library.

In this lab, you will be working on a file that contains information about MP3 files. An example is in the file **Music.txt**. It is in a very specific format. Each line of standard input contains exactly six words that describe a song that is in my MP3 library:

Title	Time (m:ss)	Artist	Album	Genre	Track

For example, here are the first ten lines of **Music.txt**:

```
Back_In_Black 4:15 AC_DC Back_In_Black Rock 6
Larks'_Tongues_In_Aspic,_Part_III 5:56 King_Crimson Three_of_a_Perfect_Pair Rock 9
Ravel,_Menuet_Antique 5:28 Casadesus,_Robert Ravel,_Complete_Piano_Music,_Disc_2 Classical 8
Pungee 3:02 Meters,_The Look-Ka_Py_Py Rock 3
Naima 4:24 Coltrane,_John Giant_Steps Jazz 6
Rachmaninoff,_Piano_Concerto_#3_in_Dm,_Opus_30,_2._Intermezzo_-_Adagio 11:43 Berman,_Lazar Rachmaninoff_-_Piano_Concerto_#3 Classical 2
Grieg,_Norwegian_Melodie,_EG_108,_#44,_Sailor's_Song_-_Hurrah_For_Jonas_Anton_Hjelm 0:32 Steen-Nokleberg,_Einar Grieg_Piano_Music,_Volume_05
Beethoven,_Sonata_#12_in_Ab,_Opus_26,_2._Scherzo._Allegro_molto 2:51 Richter,_Sviatoslav Russian_Piano_School,_V06 Classical 9
Your_Cheatin'_Heart 2:44 Armstrong,_Louis Highlights_From_His_Decca_Years,_Disc_2 Jazz 11
Pieces_Of_Dreams 5:19 Turrentine,_Stanley More_Than_A_Mood Jazz 6
```

And here's a small file for testing, **Small.txt**:

```
Countdown 2:25 Coltrane,_John Giant_Steps Jazz 3
Down_In_Brazil 6:07 Walton,_Cedar Naima Jazz 4
Giant_Steps 4:02 Puente,_Tito El_Rey Jazz 5
Giant_Steps 4:46 Coltrane,_John Giant_Steps Jazz 1
Mr._P.C. 7:02 Coltrane,_John Giant_Steps Jazz 7
Naima 4:24 Coltrane,_John Giant_Steps Jazz 6
Naima 5:16 Lyle,_Bobby Night_Breeze Jazz 5
Naima 5:36 Tjader,_Cal A_Fuego_Vivo Jazz 6
Naima 7:49 Walton,_Cedar Naima Jazz 6
Naima 8:38 Walton,_Cedar Eastern_Rebellion Jazz 2
This_Guy's_In_Love_With_You 8:10 Walton,_Cedar Naima Jazz 2
```

None of the words have spaces -- where there should be a space, there is instead an underscore.

Your job is to write the program **lib_info.cpp**, which will be called with a single command line argument:

```
UNIX> ./lib_info file
```

The argument *file* is a file in the format of **Music.txt**. You do not have to error-check this file. You may assume that it is in the correct format, and that no combination of artist/album has songs with the same track number.

The first thing your program is going to do is read in all the information and turn all underscores back into spaces.

Next, **lib_info** is going to print out all of MP3 files in the following format: For each artist (sorted lexicographically), the program will print out the artist name, followed by a colon and space, then the number of songs that have that artist's name, followed by a comma and a space, and then the total time of all songs that have that artist's name.

After each artist, you will print out each album by that artist, sorted lexicographically (and indented by eight spaces), followed again by the number of songs and total time for that album.

After each album, you will print out the title of each song on that album, sorted by track number. The format of each of these lines will be 16 spaces, the track number, a period, a space, the song's name, a colon, a space and the song's time.

Here it is on **Small.txt**:

```
UNIX> ./lib_info Small.txt
Coltrane, John: 4, 18:37
        Giant Steps: 4, 18:37
                1. Giant Steps: 4:46
                3. Countdown: 2:25
                6. Naima: 4:24
                7. Mr. P.C.: 7:02
Lyle, Bobby: 1, 5:16
        Night Breeze: 1, 5:16
                5. Naima: 5:16
```

```
Puente, Tito: 1, 4:02
        El Rey: 1, 4:02
                5. Giant Steps: 4:02
Tjader, Cal: 1, 5:36
        A Fuego Vivo: 1, 5:36
                6. Naima: 5:36
Walton, Cedar: 4, 30:44
        Eastern Rebellion: 1, 8:38
                2. Naima: 8:38
        Naima: 3, 22:06
                2. This Guy's In Love With You: 8:10
                4. Down In Brazil: 6:07
                6. Naima: 7:49
UNIX>
```

The Grading Scripts

First, you should run the grading scripts on our lab machines. If you get clever and copy enough stuff to use the grading scripts on your own machine, you'll still need to test it on our lab machines. That's how our TA's gade.

There are two programs that you should use for testing and grading. The first is **~jplank/cs302/Labs/Lab1/gradescript**. Call it with a number between 1 and 100. Each call tests your program on a different input file. The second program is **~jplank/cs302/Labs/Lab1/gradeall**. This runs all 100 tests at once.

Here's an example of doing the gradescript. Get into a fresh directory and copy your program to that directory:

```
UNIX> ls
lib_info
UNIX> ~jplank/cs302/Labs/Lab1/gradescript 1
Problem 001 is correct.

Test: ./lib_info /home/jplank/cs302/Labs/Lab1/Gradescript-Examples/001.txt
UNIX>
```

This tells you that the test was running **lib_info** on the file **/home/jplank/cs302/Labs/Lab1/Gradescript-Examples/001.txt**. Your output was correct. Yay.

The gradescript tests correctness by running my program (in **/home/jplank/cs302/Labs/Lab1**), and comparing your output with mine. The comparison is *exact*, so your output must be identical to mine, character for character. Let's see what happens when it's not.

Suppose your **lib_info** prints an extra space after the album name. Then, when you run the gradescript, it will flag an error:

```
UNIX> ~jplank/cs302/Labs/Lab1/gradescript 1
Problem 001 is incorrect.

Your standard output does not match the correct one.

TEST:

./lib_info /home/jplank/cs302/Labs/Lab1/Gradescript-Examples/001.txt

FILES:

Your standard output is in tmp-001-test-stdout.txt.
Your standard error  is in tmp-001-test-stderr.txt.

The correct standard output is in tmp-001-correct-stdout.txt.
The correct standard error  is in tmp-001-correct-stderr.txt.

Look at correct files and your files, perhaps run 'diff -y' on them, and figure out your mistake.
Please remember to delete this files when you are finished.
UNIX>
```

Usually, when this happens, the first thing that I do is take a look at the input file, and then my output and the correct output:

```
UNIX> cat /home/jplank/cs302/Labs/Lab1/Gradescript-Examples/001.txt
Pusherman 5:06 Mayfield,_Curtis Superfly Rock 2
UNIX> ./lib_info /home/jplank/cs302/Labs/Lab1/Gradescript-Examples/001.txt
Mayfield, Curtis: 1, 5:06
        Superfly: 1, 5:06
                2. Pusherman: 5:06
UNIX> /home/jplank/cs302/Labs/Lab1/lib_info /home/jplank/cs302/Labs/Lab1/Gradescript-Examples/001.txt
Mayfield, Curtis: 1, 5:06
        Superfly: 1, 5:06
                2. Pusherman: 5:06
UNIX>
```

The file is simple enough, and both outputs look identical. Dang. The next thing to do is to look at the files that gradescript creates when there is an error. There are four such files -- **tmp-001-correct-stdout.txt** and **tmp-001-correct-stderr.txt** are standard output and standard error of the correct program, and **tmp-001-test-stdout.txt** and **tmp-001-test-stderr.txt** are standard output and standard error of the erroneous program. First, list them:

```
UNIX> ls -l
total 72
-rwxr-xr-x 1 plank loci 58689 2011-08-14 14:46 lib_info
-rw-r--r-- 1 plank loci     0 2011-08-14 14:52 tmp-001-correct-stderr.txt
-rw-r--r-- 1 plank loci    87 2011-08-14 14:52 tmp-001-correct-stdout.txt
-rw-r--r-- 1 plank loci     0 2011-08-14 14:52 tmp-001-test-stderr.txt
-rw-r--r-- 1 plank loci    88 2011-08-14 14:52 tmp-001-test-stdout.txt
UNIX>
```

Neither program produces output on standard error. The sizes on the two **stdout** files differ by one. That should give you a clue. Now, run **diff -y** on them:

```
UNIX> diff -y tmp-001-correct-stdout.txt tmp-001-test-stdout.txt
Mayfield, Curtis: 1, 5:06                                         Mayfield, Curtis: 1, 5:06
        Superfly: 1, 5:06                                       |         Superfly: 1, 5:06
```

```
              2. Pusherman: 5:06                                            2. Pusherman: 5:06
UNIX>
```

diff -y prints both files and flags where they differ. Here, it's line two, which is denoted by a vertical bar. They still look the same to me. At this point, I sometimes print just the offending line to see if I see a difference:

```
UNIX> sed -n 2p tmp-001-test-stdout.txt
        Superfly: 1, 5:06
UNIX> sed -n 2p tmp-001-correct-stdout.txt
        Superfly: 1, 5:06
UNIX>
```

Dang. Still no difference. Now try piping the output to **cat -eT**, which identifies tabs and the end of each line:

```
UNIX> sed -n 2p tmp-001-test-stdout.txt | cat -et
        Superfly: 1, 5:06 $
UNIX> sed -n 2p tmp-001-correct-stdout.txt | cat -et
        Superfly: 1, 5:06$
UNIX>
```

There it is! There's an extra space at the end of the line. Now, I know from experience that y'all find the process of matching my output tedious. I agree -- however, meeting specifications is a large part of programming correctly and in teams. I won't always make you match output exactly, but you have to here.

Help and steps

You don't need to read this part if you want to try to do the lab without help or guidance. However, if you want some help, here you go: I defined classes for songs, albums and artists:

```
class Song {
  public:
    string title;
    int time;
    int track;
};

class Album {
  public:
    map <int, Song *> songs;
    string name;
    int time;
};

class Artist {
  public:
    map <string, Album *> albums;
    string name;
    int time;
    int nsongs;
};
```

You'll note, I am using these more like C structs than C++ classes -- I have defined no methods. That's ok -- there are times when you would rather do things in the manner of C, where your data is more passive, than in a strict object-oriented fashion.

I also wrote two procedures, which let me convert strings to times and back again. These store the times in single integers (seconds). I then proceeded to write the program in steps. The executables of these steps are in the lab directory, in case you want to test them against yours:

Step 1

Write the main code that reads the lines of text. Convert each time to an integer and test. I have some other test files to help:

- **Lark.txt** -- all lines with the word "Lark".
- **Rhapsody.txt** -- four renditions of "Rhapsody in Blue."
- **Dreaming.txt** -- a Dave Matthews album -- good for testing since it has a track #34.

Let's test step 1:

```
UNIX> cat Rhapsody.txt
Gershwin,_Rhapsody_In_Blue 13:22 Gershwin,_George Masters_of_the_Roll,_Volume_05 Classical 1
Gershwin,_Rhapsody_In_Blue_(Piano_Solo) 13:45 Gershwin,_George George_Gerswin_plays_Rhapsody_In_Blue_and_his_Other_Favorite_Pieces Classical 1
Gershwin,_Rhapsody_in_Blue 14:48 Finkel,_Elliot Rhapsody Classical 9
Gershwin,_Rhapsody_in_Blue 12:56 Gershwin,_George Legendary_Piano_Immortals,_Record_4 Classical 1
UNIX> ./step_1 Rhapsody.txt
13:22 = 802 = 13:22
13:45 = 825 = 13:45
14:48 = 888 = 14:48
12:56 = 776 = 12:56
UNIX>
```

Step 2

Convert all the underscores to spaces when you read words in. Then, create the song class instance when you read a song, and print it:

```
UNIX> ./step_2 Rhapsody.txt
Gershwin, Rhapsody In Blue 1 13:22
Gershwin, Rhapsody In Blue (Piano Solo) 1 13:45
Gershwin, Rhapsody in Blue 9 14:48
Gershwin, Rhapsody in Blue 1 12:56
UNIX> cat Dreaming.txt
```

```
Satellite 4:51 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 3
Jimi_Thing 5:57 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 9
Typical_Situation 5:59 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 5
Ants_Marching 4:31 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 7
Pay_for_What_You_Get 4:32 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 11
The_Best_of_What's_Around 4:17 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 1
Dancing_Nancies 6:05 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 6
Rhyme_&_Reason 5:15 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 4
Lover_Lay_Down 5:37 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 8
What_Would_You_Say 3:42 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 2
#34 4:58 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 34
Warehouse 7:06 Matthews,_Dave,_Band Under_the_Table_and_Dreaming Rock 10
UNIX> ./step_2 Dreaming.txt
Satellite 3 4:51
Jimi Thing 9 5:57
Typical Situation 5 5:59
Ants Marching 7 4:31
Pay for What You Get 11 4:32
The Best of What's Around 1 4:17
Dancing Nancies 6 6:05
Rhyme & Reason 4 5:15
Lover Lay Down 8 5:37
What Would You Say 2 3:42
#34 34 4:58
Warehouse 10 7:06
UNIX>
```

Step 3

Start dealing with artists. You should have an artist map holding all the artists. When you read in a song, you should check for the artist in the map. If it is there, print "Old Artist" and the name. If it is not, create the artist, set its name, put it into the map and print "New Artist" and the name. Test:

```
UNIX> ./step_3 Dreaming.txt
New Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
Old Artist: Matthews, Dave, Band
UNIX> ./step_3 Rhapsody.txt
New Artist: Gershwin, George
Old Artist: Gershwin, George
New Artist: Finkel, Elliot
Old Artist: Gershwin, George
UNIX> cat Lark.txt
Larks'_Tongues_In_Aspic,_Part_III 5:56 King_Crimson Three_of_a_Perfect_Pair Rock 9
Between_the_Devil_and_the_Deep_Blue_Sea 2:44 Larkins,_Ellis Jazz_Piano_III_(A_Smithsonian_Collection) Jazz 2
Exiles 5:47 King_Crimson Larks'_Tongues_In_Aspic Rock 3
Larks'_Thrak 2:37 League_of_Crafty_Guitarists Intergalactic_Boogie_Express Rock 3
Schubert-Liszt,_Hark,_Hark,_The_Lark 3:03 Paderewski,_Ignace Josef_Hofmann_&_Ignace_Jan_Paderewski_Play_Liszt Classical 5
Larks'_Tongues_In_Aspic,_Part_2 6:57 King_Crimson Larks'_Tongues_In_Aspic Rock 6
Book_of_Saturday 2:49 King_Crimson Larks'_Tongues_In_Aspic Rock 2
Schubert-Liszt,_Hark,_Hark_the_Lark 2:57 Paderewski,_Ignace Paderewski_Plays_Concert_No._1 Classical 7
Balakirev,_The_Lark 5:25 Lewin,_Michael A_Russian_Piano_Recital Classical 2
UNIX> ./step_3 Lark.txt
New Artist: King Crimson
New Artist: Larkins, Ellis
Old Artist: King Crimson
New Artist: League of Crafty Guitarists
New Artist: Paderewski, Ignace
Old Artist: King Crimson
Old Artist: King Crimson
Old Artist: Paderewski, Ignace
New Artist: Lewin, Michael
UNIX>
```

Step 4

Do the same with the album in the artist's class. Check to see if the album is there. If so, print out its name. If not, create it and print out its name. Test:

```
UNIX> ./step_4 Dreaming.txt
New Artist: Matthews, Dave, Band
New Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
Old Artist: Matthews, Dave, Band
Old Album: Under the Table and Dreaming
UNIX> ./step_4 Lark.txt
New Artist: King Crimson
New Album: Three of a Perfect Pair
New Artist: Larkins, Ellis
New Album: Jazz Piano III (A Smithsonian Collection)
Old Artist: King Crimson
New Album: Larks' Tongues In Aspic
New Artist: League of Crafty Guitarists
New Album: Intergalactic Boogie Express
New Artist: Paderewski, Ignace
New Album: Josef Hofmann & Ignace Jan Paderewski Play Liszt
Old Artist: King Crimson
Old Album: Larks' Tongues In Aspic
Old Artist: King Crimson
Old Album: Larks' Tongues In Aspic
Old Artist: Paderewski, Ignace
New Album: Paderewski Plays Concert No. 1
New Artist: Lewin, Michael
New Album: A Russian Piano Recital
UNIX>
```

Step 5

Insert the songs into the albums' song lists. Write code to traverse the artists/albums after you're finished reading the information from the file. Just print out names -- don't worry about songs or times yet. Test:

```
UNIX> ./step_5 Dreaming.txt
Matthews, Dave, Band
        Under the Table and Dreaming
UNIX> ./step_5 Rhapsody.txt
Finkel, Elliot
        Rhapsody
Gershwin, George
        George Gerswin plays Rhapsody In Blue and his Other Favorite Pieces
        Legendary Piano Immortals, Record 4
        Masters of the Roll, Volume 05
UNIX> ./step_5 Lark.txt
King Crimson
        Larks' Tongues In Aspic
        Three of a Perfect Pair
Larkins, Ellis
        Jazz Piano III (A Smithsonian Collection)
League of Crafty Guitarists
        Intergalactic Boogie Express
Lewin, Michael
        A Russian Piano Recital
Paderewski, Ignace
        Josef Hofmann & Ignace Jan Paderewski Play Liszt
        Paderewski Plays Concert No. 1
UNIX> ./step_5 Small.txt
Coltrane, John
        Giant Steps
Lyle, Bobby
        Night Breeze
Puente, Tito
        El Rey
Tjader, Cal
        A Fuego Vivo
Walton, Cedar
        Eastern Rebellion
        Naima
UNIX>
```

Step 6

Print out the songs too, and print them out in the correct format, with track number and time:

```
UNIX> ./step_6 Rhapsody.txt
Finkel, Elliot
        Rhapsody
                9. Gershwin, Rhapsody in Blue: 14:48
Gershwin, George
        George Gerswin plays Rhapsody In Blue and his Other Favorite Pieces
                1. Gershwin, Rhapsody In Blue (Piano Solo): 13:45
        Legendary Piano Immortals, Record 4
                1. Gershwin, Rhapsody in Blue: 12:56
        Masters of the Roll, Volume 05
                1. Gershwin, Rhapsody In Blue: 13:22
UNIX> ./step_6 Lark.txt
King Crimson
        Larks' Tongues In Aspic
                2. Book of Saturday: 2:49
                3. Exiles: 5:47
                6. Larks' Tongues In Aspic, Part 2: 6:57
        Three of a Perfect Pair
```

```
                9. Larks' Tongues In Aspic, Part III: 5:56
Larkins, Ellis
        Jazz Piano III (A Smithsonian Collection)
                2. Between the Devil and the Deep Blue Sea: 2:44
League of Crafty Guitarists
        Intergalactic Boogie Express
                3. Larks' Thrak: 2:37
Lewin, Michael
        A Russian Piano Recital
                2. Balakirev, The Lark: 5:25
Paderewski, Ignace
        Josef Hofmann & Ignace Jan Paderewski Play Liszt
                5. Schubert-Liszt, Hark, Hark, The Lark: 3:03
        Paderewski Plays Concert No. 1
                7. Schubert-Liszt, Hark, Hark the Lark: 2:57
UNIX>
```

Step 7

Let's worry about the album total time. When you're reading in a song, update the album's time. Make sure you set the album's time to zero when you creat the album. Test it by printing out the correct line for each album. Note, you don't need to keep track of the number of songs, because you have that information already in the **songs** map. Test:

```
UNIX> ./step_7 Rhapsody.txt
Finkel, Elliot
        Rhapsody: 1, 14:48
                9. Gershwin, Rhapsody in Blue: 14:48
Gershwin, George
        George Gerswin plays Rhapsody In Blue and his Other Favorite Pieces: 1, 13:45
                1. Gershwin, Rhapsody In Blue (Piano Solo): 13:45
        Legendary Piano Immortals, Record 4: 1, 12:56
                1. Gershwin, Rhapsody in Blue: 12:56
        Masters of the Roll, Volume 05: 1, 13:22
                1. Gershwin, Rhapsody In Blue: 13:22
UNIX> ./step_7 Dreaming.txt
Matthews, Dave, Band
        Under the Table and Dreaming: 12, 62:50
                1. The Best of What's Around: 4:17
                2. What Would You Say: 3:42
                3. Satellite: 4:51
                4. Rhyme & Reason: 5:15
                5. Typical Situation: 5:59
                6. Dancing Nancies: 6:05
                7. Ants Marching: 4:31
                8. Lover Lay Down: 5:37
                9. Jimi Thing: 5:57
                10. Warehouse: 7:06
                11. Pay for What You Get: 4:32
                34. #34: 4:58
UNIX>
```

The final step:

Write the code to update the artist's song count and total time when you read in a song. Then print out the correct line for each artist. Test, comment, and you're done.