CS 392: Homework Assignment 3
Due: March 8, 11:55pm

Philippos Mordohai
Department of Computer Science
Stevens Institute of Technology
Philippos.Mordohai@stevens.edu

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment.

Late Policy. No late submissions will be allowed without consent from the instructor. If urgent or unusual circumstances prevent you from submitting a homework assignment in time, please e-mail me explaining the situation.

Objective

In this assignment, we will work on opening, reading from and writing to files.

Movielens Datasets

Data for this assignment can be found in the accompanying zip file. You should use the following datasets for all problems:

(1) 100,000 ratings from 1000 users on 1700 movies. Released 4/1998.

(2) 1 million ratings from 6000 users on 4000 movies. Released 2/2003.

(3) 10 million ratings and 100,000 tag applications applied to 10,000 movies by 72,000 users. Released 1/2009.

u.data: Data are stored as strings separated by a tab (`\t`) as follows: `user_id    item_id      rating   timestamp` For example:

```
196 242 3 881250949
186 302 3 891717742
22  377 1 878887116
244 51  2 880606923
166 346 1 886397596
```

```
298 474 4 884182806
115 265 2 881171488
253 465 5 891628467
```

Problem 1. text2bin (25 points)

Create a program that transforms the data file from text to binary. Your program should be called text2bin and take two mandatory command line arguments:
`text2bin <input filename> <output filename>`
The input filename is the u.data file in the dataset. The output filename is the file to store the binary output of your program. The output should include the same data in the same order but in binary using the following format:

```
user_id         item_id         rating          timestamp
4-byte integer  2-byte integer  1-byte integer  8-byte integer
```

Your code could include the following steps:

(1) You **must** use `fgets()` to read the data.

(2) Use `fwrite()` to produce the output.

(3) To read the four strings from the input file: use `sscanf()`, `strtok()`, or `strtoul()` and similar functions.

`strtoul()`:
https://www.tutorialspoint.com/c_standard_library/c_function_strtoul.htm

`strtok()`:
https://www.tutorialspoint.com/c_standard_library/c_function_strtok.htm
Note that the first call to `strtok` must pass the C string to tokenize, and subsequent calls must specify NULL as the first argument, which tells the function to continue tokenizing the string you passed in first.

Problem 2. bin2text (25 points)

Create a program that transforms the binary file you created back to text. Your program should be called bin2text and take two mandatory command line arguments:
`bin2text <input filename> <output filename>`
The input filename is the binary file. The output filename is the file to store the text output of your program. The output should be identical to the original dataset file.

Your code could include the following steps:

(1) Use any of the character/string/buffer C stream input functions to read the data.

(2) Use `fprintf()` to produce the output.

Problem 3. bin2indexed (40 points)

In addition to u.data used above, we will now use u.item as well. u.data includes item IDs for movies but looking up the actual title of the movie in u.item is slow. Create a program that replaces the item ID in the binary file with the position (offset) of the corresponding movie item in the u.item file. Your program should be called bin2indexed and take three mandatory command line arguments:
`bin2indexed <binary file> <item file> <output filename>`
The binary file is the file produced by text2bin. The item file is u.item from the zip file. The output filename is the file to store the binary output of your program. The output should include the same data in the same order but in binary using the following format:

```
user_id        item_file_offset rating         timestamp
4-byte integer 8-byte integer   1-byte integer 8-byte integer
```

u.item contains one line per movie. Movies are sorted by item ID. For example:

offset	line in u.item
0	1 | Toy Story (1995) | 01-Jan-1995 || http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)...
124	2 | GoldenEye (1995) | 01-Jan-1995 || http://us.imdb.com/M/title-exact?GoldenEye%20...

Your code could include the following steps:

(1) `ftell()` can be used to obtain the current offset of a stream within a file. For example: it should be zero right after the file is opened.

(2) You may need to use `malloc()/free()/realloc()` to dynamically allocate memory for the index. Do not assume that you know how many movies there are. If this turns out to be too challenging, peek at the total number of movies for a small penalty.

Time your programs (10 points)

Use `time -p <command + arguments>` to time your programs with the various datasets. Include timing results for all three programs on each of the three datasets in your report. How does your program scale with file size?

Assumptions

(1) The input files will be valid. You do not need to verify the format, only that the files exist.

Requirements

(1) Your program should compile and work correctly. Compiling and performing part of the requirements is better than not compiling.

(2) Use comments where necessary to explain what you are doing. No comments or over-commenting are both bad.

(3) All files must be named **exactly** as described below.

(4) Do not leak memory. Free all allocated memory and close all opened files.

Deliverables

(1) text2bin.c

(2) bin2text.c

(3) bin2indexed.c

(4) A pdf file with brief explanations of your approach and timing results.