CSE225 Data Structures, 2014-2015 (FALL)

PROJECT #1 (Due October 20, 2014, MONDAY, 01:00 a.m.)

This project is a programming assignment in C which builds a review-score list based on the user's reviews collected from WWW. These reviews are in some basic categories according to the scores they have. This program will let the user add, remove, update the review-score list.

The data structure that you use is as detailed below:

1. The review-score number,
2. The number of the total reviews on that review-score list
3. A list of the reviews in that review-score. For each review, you should keep:
    - The id of the review
    - The length of the review (how many words are included in that review)
    - The polarity of the review(if it is a positive review, negative review or a neutral review)

To hold the list of all review-scores, you will use ***a master-linked-list***. This linked list will exactly contain as many nodes as the number of different review-scores where each node *i* represents the score a_i in the review-score list. The node *i* should contain the information of a_i such as *the review-score-number and the number of total reviews under the review-score number category.* Each node i in this linked list will also contain ***a pointer field***. This pointer will function as the header of a linked list LL(a_i) that will list the reviews in the relevant review-score a_i. The size of LL(a_i) will depend upon the number of the reviews in a_i. which differs from node to node that will list the reviews in the relevant review-score-number a_i. The size of LL(a_i) will depend upon the number of the reviews in a_i. Each node *i* in LL(a_i)will consist of

- The id of the review
- The length of the review (how many words are included in that review)
- The text of the review
- The polarity of the review(if it is a positive review, negative review or a neutral review)

Your system will have the following functionalities; the details of these functionalities are given below:

1. Add a new review-score node into the master-linked list.
2. Add also some reviews into a specific review-score category.
3. Show the average score of all of the reviews.

4. Show the review id s and the total number of them in a specific polarity (positive, negative or neutral)
5. List the id s of reviews which have a particular word.
6. Remove the reviews between in a specific score-scope.

The Details of These Functionalities:

1. This function adds a review-score category to the master-linked-list whose score-number is specified as parameter. The master-linked-list is in ascending order according to score numbers. In this function, the review list is not specified; the reviews listed under that score will be added later. In this system, the numbers of the scores are unique. Thus, if the user attempts to add a score-node into the master-linked list with an existing score-number, you should not allow this operation and give a warning message.

2. This function adds a review to the review list of a score-number node. For that, the score number to which the review is added, the id of the review, the length (in words), the text of the review and the polarity of the review are specified as parameters. In this function, you should consider the following issues:
• If the score with a specified number does not exist in the master-linked-list, you should not allow the operation and give a warning message.
• In this system, all id s of the reviews are unique. Thus, if the user attempts to add a review with an existing id, you should not perform the operation and give a warning message.

3. This function shows the average score of all of the reviews. You need to traverse all score nodes including their review-list and calculate the average as follows:

$$S_{\text{mean}} = \frac{R}{N}$$

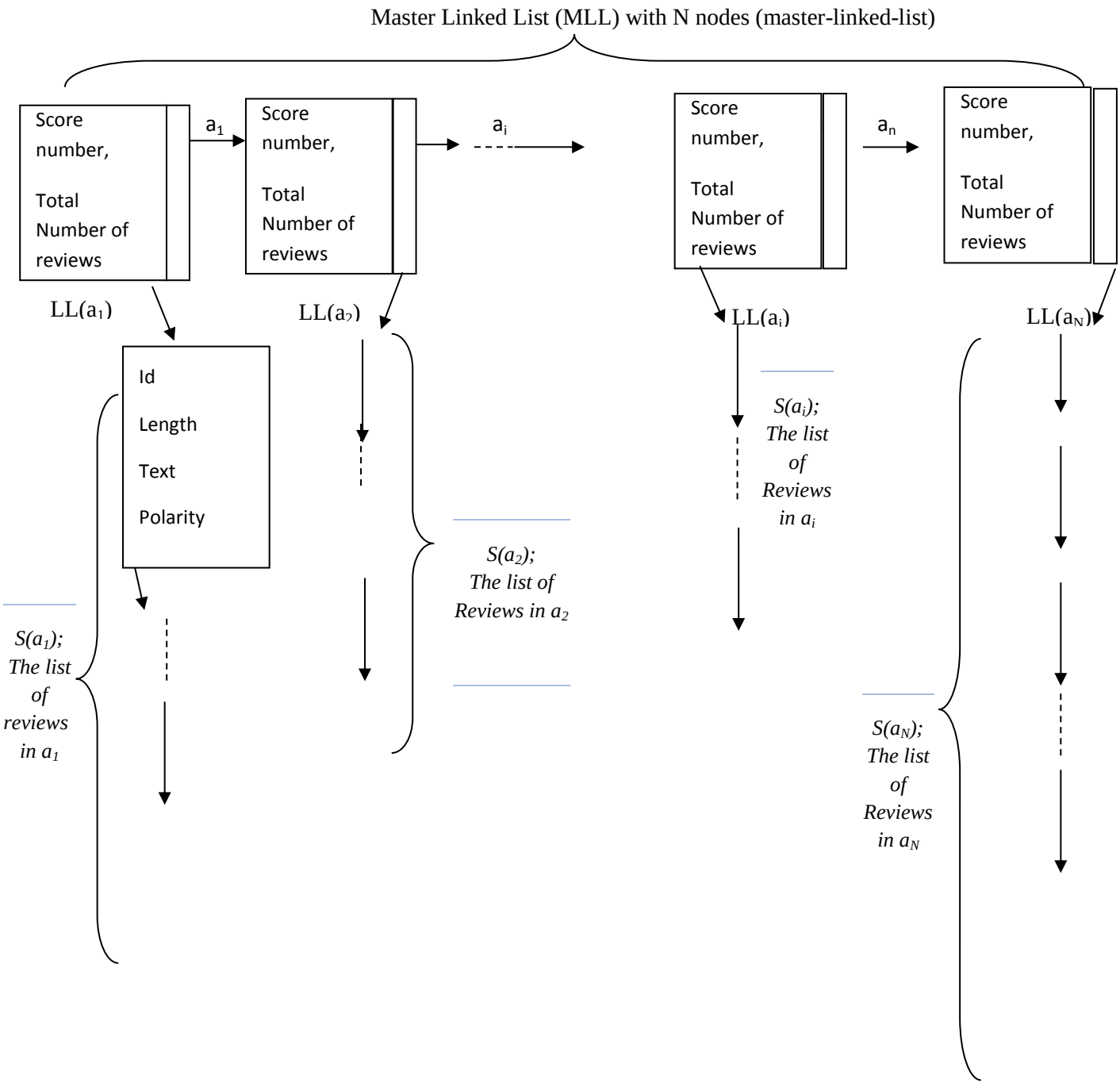
where R is the total review-score calculated from all of the reviews from all of the review-score numbers, and N is the total number of reviews.

4. In this function you should list all reviews with a specified polarity. The polarity of the review is specified as parameter by the user. In order to find the polarity of review you should keep track of the words: PROS, CONS. If a review contains PROS then its polarity is positive, if it contains CONS then its polarity is negative, or if it includes both PROS and CONS or none of them then it means that its polarity is neutral. The output should include the id s of the reviews and the total number of them. Note that if there is no such a review, you should display ---none--- .

5. In this function you should list all reviews with a specified word. The word is specified as parameter by the user. The output should include the id s of the reviews. Note that if there is no such a review, you should display ---none--- .

6. This function removes the specified score node including all of its reviews from the master-linked list, whose score-number is specified as a parameter. If this score-number does not exist in the linked-list, you should not allow this operation and give a warning message.

SCHEMA OF THE MASTER-LINKED-LIST AND THE LINKED-LIST OF REVIEWS:



In your demo, we will run your program by **selecting each of above function-options with a different set of input files.** We want to see if they are **working correctly or not**, **and your demo grade will be calculated based on number of your functions which are working correctly.**

Of course, other questions based on your implementation and coding structure will be asked you. These questions will be those kinds of questions which could be answered by only the students who really implement his/her project.

If your file could not be compiled then you will get zero, unfortunately.

The main goal of this project is to be familiar with linked-list. So, if you use arrays instead of linked-lists then you will get zero, unfortunately.

In this project you are expected to develop an algorithm that is capable of finding a solution to the above problem and ***implement this algorithm in ANSI C that runs under UNIX***. (You can also use cygwin or wagrant in order to compile and run your codes.)

You are responsible for demonstrating your program to your TA Berna Altinel on the scheduled day that will be announced later.

CODE SUBMISSION:

You should use the following email address in order to submit your code:
datastr.mufe at gmail dot com

Your any submission after the project submission due date, will not taken into consideration.

You are required to exhibit an ***individual effort*** on this project. Any potential violation of this rule will lead everyone involved to **failing from the course** and necessary disciplinary actions will be taken.

Good luck!!!