Project 1

Instagram Trending

C S 105C Spring 2022

Problem Statement:

Instagram is re-doing its Internet stories trending feature, and you are asked to develop a prototype by means of a C++ program that will display a list of the top Internet stories.

Program Requirements:

1. The program must display the stories that have a popularity **score** which is equal to the **statistical mode**. Why not the largest scores? Who knows?

The **mode** of a set of numbers is the number that occurs most often (with the greatest frequency). To find the mode, it is best to put the numbers in order (sort them) so that *repeated* numbers are next to each other. Then, **count how many** there are of each number. The number that appears most often (the highest count) is the **mode**.

For more help calculating the mode, look here:

https://www.mathsisfun.com/mode.html#:~:text=To%20find%20the%20mode%2C%20or,most%20often%20is%20the%20mode.

2. If the data has **no mode** (i.e., none of the 'score' values occurs more than once), then the program must display the first five stories read from the input file. If **there is** a mode, notice that you won't necessarily know *how many stories* have the mode. (*Is this true?*)

3. The data about the stories is in an **input** file that contains the following items on separate lines (see #5 below):

title (a sequence of numbers and/or letters, it may contain spaces in it)

url (a regular URL, like "http://www.twitter.com/story1", **without** any spaces in it)

score (an integer number, greater than zero)

4. The **output** produced by the program must look like:

"Mode: " <mode | "No mode was found.">

<blank line>

<Story 1 title>

<Story 1 URL>

<blank line>

<Story 2 title>

<Story 2 URL>

Etc.

See the sample output produced for the sample input mentioned below.

5. The program must first read the stories data from a text input file into an **array of structures**. The input file contains:

- First, the number of stories found in the file, and
- Then, the data items for each story, in the order listed above, with each data item on a separate line (3 lines per story).

I have provided a sample input file, named "**stories**" (make sure any file you use is a text file—for example, create it with a text editor like **vi** on Linux). Find this file in the assignment.

Note: Ask the user to type the **name of the file** they want to be read.

6. The program must **dynamically allocate memory** for the array to store only the number of stories found in the input file. You must use dynamic memory allocation to achieve this feature. Demonstrate your pointer prowess by using **pointer notation** instead of **array notation** in the program.

7. Write a function to read the input data into the dynamic array. Where do you declare this array? What parameters should this function receive? What should the function return and of what data type is it?

8. Write a function to display the stories to the screen, given the mode value.

9. Write a function to find the mode in the data, call it **findMode**, and have it accept as parameters:

- a pointer to the array of structures,
- an integer that indicates the number of elements in the array.

This function must determine the mode of the data set (using each story's **score**). The **mode** is the score value that the function must return.

If the data set has no mode, the function must return -1.

If there are two or more modes (i.e., two or more scores occur with the same frequency), the function must return the **first** mode found in the input file.

Additional Requirements:

1. The program must be done in a **Linux** environment, using the command line compiler **g++**. Do not only use Code::Blocks, Eclipse, or other IDEs to compile.
2. The program must compile and run, otherwise you will receive a grade of 0.
3. The program must be **modular** (use top-down design), with significant work done by functions. Each function must perform a single, well-defined task. Are there any additional functions that you envision in your design?
4. Follow the *Style Guidelines* document on the Canvas *Project Information* page. Specially, pay attention to the comments required for the top of the file and for each function definition. Points will be deducted if the program violates the style guidelines. For more details on penalties, follow the *Submission Policy* document on the same Canvas page.

Grade Breakdown:

Design, 25%

- Describe your data structures and algorithms (not meant as "C++ Algorithms") in a separate text file
- Explain any design choices you decided to make
- Add instructions on how to compile and run the program on the GDC lab machines

Source code, 50%

- The source code (as one .cpp file) follows the style guidelines (headers, comments, etc.)
- There is a "log" of who worked on each function, in their headers (re: pair programming)
- The program compiles and links without errors on the GDC lab machines

Test cases and output, 25%

- The program runs as expected
- The output is formatted exactly as shown in the sample file **project1output.pdf**
- The screen output of **five** good test cases (different input files) is correct and is

attached in a separate file (as screenshots, NOT as text files)

Submission Logistics:

Submit all your files (see underlined files above) separately. We will figure out your UT EIDs from your group's submission.

Only one submission set is necessary per group.

Submit all files via the Canvas “Project 1” assignment.