Prof. Victor Matos

CS1-Fall2021
Midterm 2 - C++ Programming - Rate My Products App

An online retailer wants to summarize the ratings of their products as noted by the customers' supplied evaluations. The rating is based on a five-stars system, where five is the highest mark a product may receive (excellent), and one star is the lowest mark (poor ranking). Currently, the retailer sells ten different items. However, they expect to add new products soon to their lineup.

In order to solve the problem your app will create and use a parallel-array based database (the arrays are described below). The text file **c:/temp/rateMyProduct.txt** is a compilation of the customers' supplied ratings. Each disk record represents the customer's review of a single product. A record contains a product ID and a rating. The product ID is currently a number between 1 and 10 and the rating is a value between 1 and 5. So, for example, the input record
7, 4
means that a customer has ranked item 7 as a 4 stars product. Keep in mind that not necessarily each product has been rated.

You are asked to write an app that,

1. Creates a parallel-array database using supplied disk data and calculated values.
2. Summarizes the input data and reports the average score of each product.
3. Identifies and displays the product(s) with the highest and lowest marks.
4. Allows the operator to iteratively retrieve by product ID the statistics of a given product.

Appendix A shows a sample of the output format that your app should produce. In addition, the app MUST include the following functions.

```
//Prototypes
void populateArrays(int n, int aId[], int aSum[], int aCount[], string fileName);
void calculateProductAvg(int n, int aId[], int aSum[], int aCount[], double aAvg[]);
void showAllRaitings(int n, int aId[], double aAvg[], int aCount[]);
void showBestWorstProducts(int n, int aId[], double aAvg[]);
void searchAndShowSelectedProduct(int n, int aId[], double aAvg[], int aCount[]);
```

where,

Entity	Meaning
aId[i]	Each cell holds the product id number. aId[1]=1, aId[2]=2, …, aId[10]=10.
aSum[i]	Keeps the sum of all stars given to the product aId[i] by all customers reviewing the item.
aCount[i]	Stores the count of customers who made a review of item aId[i].
aAvg[i]	Is the average rating of product aId[i] calculated as (aSum[i] / aCount[i]) (assuming that aCount[i] is not zero).
fileName	String identifying the input dataset "**c:/temp/rateMyProduct.txt**"

The following section describes the purpose of each method.

```
void populateArrays(int n, int aId[], int aSum[], int aCount[], string fileName);
```

This function is responsible for reading the disk file whose name is supplied in the parameter *filename*. The arrays aId, aSum, and aCount must be populated with the disk file. As an example, consider product 3. Assume there are altogether 4 reviews, each giving 3, 3, 5, and 2 stars, respectively. Therefore, aId[3] = 3, aSum[3] = 3+3+5+2 = 13, and aCount[3] = 1+1+1+1 = 4. The parameter n tells the number of different products under evaluation. For convenience, you may consider that product 0 doesn't exist (or is not being reviewed). Hence, array entries for this product should be gaId[0] = 0, aSum[0] = 0, and aCount[0] = 0.

```
void calculateProductAvg(int n, int aId[], int aSum[], int aCount[], double aAvg[]);
```

For each rated product you need to calculate its corresponding average number of stars. For example, consider again the case of product 3 which has been rated by four customers. Assume, aId[3] = 3, aSum[3] = 13, and aSum[3] = 4. The average number of stars aAvg[3] = 3.2 which is calculated as (aSum[3] / aCount[3]) = 13 / 4 = 3.2 (must be displayed with only one decimal position.)

```
void showAllRaitings(int n, int aId[], double aAvg[], int aCount[]);
```

Displays summary data for each product. An output line contains the product ID, average number of stars, and total number of received reviews. For instance, product number 3 is reported as

```
Prod.  3        Stars: 3.2      Reviews: 4
```

Observe that a similar output needs to be shown for each product (see Appendix A).

EXTRA CREDITS. You may also calculate the standard deviation of the sample (see Appendix C). In that case the function's header should be

```
void showAllRaitings(int n, int aId[], double aAvg[], double std[], int aCount[]);
```

```
void showBestWorstProducts(int n, int aId[], double aAvg[]);
```

Scan the array aAvg[] to find out which are the best and the worst rating. Observe that multiple products may share the best/worst average.; in that case, each product must be reported (see Appendix A).

```
void searchAndShowSelectedProduct(int n, int aId[], double aAvg[], int aCount[]);
```

Provides an interface for the user to enter a product number repeatedly. The app uses that number to search the database and report its corresponding stats. If the product is not found, the app says, "Sorry – Product not found – Try again". Otherwise, it shows the product's average number of stars and its number of reviews. The output must be formatted as indicated on the final portion of Appendix A. When the supplied product number is 0 (zero), the cycle ends.

Design Specs.
You have to write the app so that your main() function is the first method to be listed in the report. User-defined functions must appear after the main() function. Your main() function must call the supporting functions listed above. Only your own functions are allowed; you must not use any included sort/search library function. Average rating must be displayed using only one decimal position.

Hints.
Consider using the function **getline(source, string, ',')** to read a value from a stream source and copy it into a string variable. Remember that the function **stoi(string)** converts a string value into its integer representation. The provided disk file (**rateMyProducts.txt**) was created in a Windows machine; therefore, each line ends on the character '**\n**'.

APPENDIX A - Sample Console Input/Output

```
Dataset - Product Rating (Five Stars)
Prod.  1         Stars: 3.2      Reviews: 16
Prod.  2         Stars: 4.0      Reviews: 13
Prod.  3         Stars: 3.2      Reviews: 4
Prod.  4         Stars: 4.2      Reviews: 11
Prod.  5         Stars: 3.1      Reviews: 8
Prod.  6         Stars: 3.9      Reviews: 7
Prod.  7         Stars: 2.2      Reviews: 10
Prod.  8         Stars: 4.0      Reviews: 9
Prod.  9         Stars: 3.0      Reviews: 13
Prod. 10         Stars: 3.9      Reviews: 9
-------------------------------------------------
Best Product(s):  Product 4(4.2 stars),

Worst Product(s): Product 7(2.2 stars),
-------------------------------------------------

Retrieve product by ID

Enter product ID [0 to end]: 4
Product 4 (4.2 Stars, 11 Reviews)

Enter product ID [0 to end]: 7
Product 7 (2.2 Stars, 10 Reviews)

Enter product ID [0 to end]: 15
Sorry - Product not found - Try again

Enter product ID [0 to end]: 0

All done!
```

APPENDIX B - Contents of the input file rateMyProduct.txt (Each row contains Product ID and stars given by a customer)

```
2, 4
1, 5
5, 4
9, 3
5, 1
6, 3
2, 3
3, 3
7, 2
5, 3
4, 4
2, 3
6, 4
2, 5
3, 3
10, 2
4, 3
4, 5
2, 3
9, 3
5, 3
8, 4
4, 3
9, 2
6, 2
9, 2
1, 3
5, 4
7, 1
1, 5
1, 1
7, 2
4, 5
4, 5
7, 1
7, 2
2, 5
10, 3
8, 5
3, 5
2, 5
10, 5
1, 3
7, 4
7, 2
6, 5
1, 5
8, 5
3, 2
1, 2   (continue on left column)
```

```
7, 1
8, 2
2, 4
1, 2
5, 2
1, 3
2, 4
8, 4
4, 2
10, 5
9, 3
7, 2
1, 4
9, 1
2, 4
1, 5
5, 3
9, 4
6, 3
1, 2
7, 5
1, 2
2, 5
9, 5
2, 5
10, 5
1, 4
8, 4
4, 5
8, 3
8, 5
10, 3
2, 2
10, 3
4, 5
5, 5
10, 4
6, 5
4, 4
4, 5
9, 4
1, 2
9, 2
10, 5
8, 4
9, 3
9, 5
1, 3
9, 2
6, 5
```

Appendix C. The standard deviation of a sample { x_1, x_2, … x_N } is defined as

$$std = \sqrt{\frac{\sum(x_i - avg)^2}{N}}$$

Where N is the size of the population, avg is the average of the sample, and x_i is each element from the sample.

ECC CS1 – Spring 2022
Midterm 2 – Prof. V. Matos

First Name _____ Last _____

Exclusive Union

Assume v1 and v2 are vectors holding integer values (vector<int>). As an example, consider

```
vector<int> v1 { 44, 22, 33, 11, };
vector<int> v2 { 22, 33, 77 };
```

You are asked to write the function

```
vector<int> exclusiveUnion(vector<int> v1, vector<int> v2);
```

to calculate the result of the **exclusive union** of the incoming vectors v1 and v2. The function must return a vector<int> holding the solution.

> The exclusive union of the vectors v1 and v2 is the combination of values that exclusively belong to v1 and v2, respectively. Observe that common elements of v1 and v2 are not included in the result.

For example, in the sample above, 22 and 33 are common to v1 and v2. Hence, they will not be part of the answer. On the other hand, observe that the values 44 and 11 are exclusively found in v1 but not in v2. Consequently, 44 and 11 will appear in the result. Finally, 77 is an exclusive value of v2, and we will include it in the answer. Therefore, for this sample, the function will return a solution vector holding {**44**, **11**, 77}.