CSCI 1300 CS1: Starting Computing
Ashraf, Cox, Spring 2020
Homework 6
Due: Saturday, March 7, by 6 pm
(5 % bonus on the total score if all three parts are submitted by 11:59 pm March 6)

Objectives

- Understand file IO

You can find **hw6 note: file IO** on Moodle.

Submissions

- **Conceptual reviews(mcq)**. There are a few multiple-choice questions to check your conceptual understanding. Don't forget to complete them!
- **C++ files**. All files should be named as specified in each question, and they should compile and run on Cloud 9 to earn full points. TAs will be grading styles of your code and comments. Please see the style guide on Moodle and the summary note on Moodle. At the top of each file, write your name with the following format:

```
// CS1300 Spring 2020
// Author: Punith Sandhu
// Recitation: 123 – Favorite TA
// Homework 6 - Problem # …

/*
 * This function converts a temperature in fahrenheit to celsius
 * and prints the equivalence.
 * Parameters: fahrenheit - degrees fahrenheit
 * Return: equivalent temperature in celsius
*/

double fahrenheit_to_celsius(double fahrenheit){
      double celsius = (fahrenheit - 32) * (5/9.0);
      return celsius;
}
```

  For each question, create a program that calls the function in the main. Here is an example.

- **Code runner**. Your program will be graded by the code runner. You can modify your code and re-submit (press Check again as many times as you need to, up until the assignment due date.

Questions

Question 1(10pt): Item list

We have a list of products and their prices. Each line contains the name of the item and its price separated by a comma. Write a program that reads a txt file and computes the following:

- The number of lines in the file
- The most expensive product
- The least expensive product

Edge cases:

- If multiple items qualify as the most/least expensive product, then print only the first product.
- The files have not been pre-processed. It might contain empty lines in the file.
- If the file cannot be opened, then print "`Could not open file.`"

Sample file(**products.txt**):

```
Apple EarPods, 19.99
Bose QuietComfort 35, 292.42

Apple AirPods , 139.00
Beats Solo3, 189.98
```

Sample run1 (**bold** is user input)

```
Enter the file name:
products.txt
The number of lines: 4
The most expensive product: Bose QuietComfort 35
The least expensive product: Apple EarPods
```

Sample run 2 (**bold** is user input)

```
Enter a filename:
does-not-exist.txt
Could not open the file.
```

The file should be named as `itemList.cpp`. Don't forget to head over to the code runner on Moodle and paste your solution in the answer box!

Question 2(15pt): Sales data

A company sells software products.It keeps a record of the number of products that were sold by an employee for a particular month in a calendar year. Write a function, **readSales** to read the txt file and store the name and sales in two separate arrays. The function returns the number of employees stored.

Function specifications:

- The function name: **readSales**
- The function parameters (in this order):
  - A file name of type `string`
  - An array of type `string` to store names of employees
  - A 2D `integer` array with 12 columns to store sales of employees
  - The size of the array as an `int`
- The function returns an integer depending on the following conditions:
  - It returns -1 if the file cannot be opened.
  - It returns the number of employees stored in the array if the function can open and read the file
  - It returns the size of the array (the capacity of the string array or the number of rows in the 2D array) if the arrays are full.

Sample file (**sales.txt**):

```
Vipra
60, 80, 50, 70, 85, 32, 94, 81, 45, 65, 91, 82
Chandan
1, 25, 18, 22, 23, 16, 5, 33, 21, 27, 23, 102
Malvika
55, 54, 52, 56, 58, 52, 51, 59, 58, 50, 54, 55
```

Sample run 1 (**bold** is user input)

```
Enter the file name:
sales.txt
Returned value (number of employees): 3
Names[0] = Vipra
sales[0][0] = 60
sales[0][1] = 80
sales[0][2] = 50
sales[0][3] = 70
sales[0][4] = 85
sales[0][5] = 32
sales[0][6] = 94
sales[0][7] = 81
sales[0][8] = 45
```

```
sales[0][9] = 65
sales[0][10] = 91
sales[0][11] = 82
```

The file should be named as `sales.cpp`. Don't forget to head over to the code runner on Moodle and paste only your function in the answer box! In the main, make sure that you call the function and output the return value.

Extra Credit 5 pt: in the main, print out the average scores of hmwk for each student.

Sample run 2 (**bold** is user input)

```
Enter the file name:
sales.txt
Returned value (number of employees): 3
Vipra: 69.58
Chandan: 26.33
Malvika: 54.50
```

Question 3(15pt): College Admission Algorithm

As you know, the college admissions process involves a lot of types of data from prospective students to make decisions. With the number of applicants increasing, colleges rely on algorithms to select applicants. An algorithm could use quantitative data--such as GPA and SAT score--to provide recommendations. In fact, there is more data available than ever. Many colleges even track data about prospective student engagement - e.g., whether they open emails, visit the college website, engage on social media, etc. This creates a "demonstrated interest" value.

Based on a recent survey of college admissions officers, we know some of the weights that humans tend to give to these different types of data. Your task will be to write a program that iterates through a list of data points and provides a recommendation (using an algorithm described later in the question) for which prospective students are likely to be the best candidates for admission.

Prospective student data is organized in a text file such that the data for each student is on one line, with the values separated by commas. Here is an example of a text file containing three students' data:

Sample File

```
Student,SAT,GPA,Interest,High School Quality,Sem1,Sem2,Sem3,Sem4
Abbess Horror,1300,3.61,10,7,95,86,91,94
Adele Hawthorne,1400,3.67,0,9,97,83,85,86
Adelicia von Krupp,900,4,5,2,88,92,83,72
```

The data includes (in order):
Student: a unique identifier for each student, i.e. their string name (type `string`)
SAT score: value between 0 and 1600 (type `int`)
GPA: value between 0 and 5 (type `double`)
Interest: value between 0 and 10, from very low to very high interest (type `int`)
High School Quality: value between 0 and 10, from very low-quality to very high-quality high school curriculum (type `int`)
Sem1: average grade for semester 1 (type `int`)
Sem2: average grade for semester 2 (type `int`)
Sem3: average grade for semester 3 (type `int`)
Sem4: average grade for semester 4 (type `int`)

To determine the students that are likely to be the best candidates for admission, an overall score needs to be calculated for each student based on the following weights: 40% GPA, 30% SAT, 20% strength of curriculum, 10% demonstrated interest. To make this work, you will also need to normalize both GPA and SAT so that they are also on a 0 to 10 scale. To do this, multiply the GPA by 2, and divide the SAT score by 160. **All students who have a score of 6 or higher are chosen** (this is the **score criteria** for choosing a student). For example, for a student with a 1300 SAT score, a 3.61 GPA, 10 out of 10 for interest and 7 out of 10 for high school quality, the overall score is calculated as:
((3.61 * 2) * 0.4) + ((1300 / 160) * 0.3) + (10 * 0.1) + (7 * 0.2) = 7.73 out of 10. Since 7.73 >= 6, this student is chosen.

An algorithm with just this criteria might systematically miss certain kinds of edge cases. For example, what if a student has a 0 for demonstrated interest because they don't use social media or have access to a home computer? What if a student has a very high GPA but their SAT score is low enough to bring their score down; could this mean that they had a single bad test taking day?

Therefore, the algorithm should also check for certain kinds of outliers. It should check for: **(1)** demonstrated interest scores of 0 and **(2)** a normalized GPA that is more than 2 points higher than the normalized SAT score. **If either of these conditions is true and the student has a score of 5 or higher, the student is chosen** (this is the **outlier criteria** for choosing a student).

Finally, consider the importance of an algorithm being able to flag students who might have a lower overall GPA but have shown improvement over time. Therefore, **if the score of each semester is higher than each previous semester and the student has a score of 5 or higher, the student is chosen** (this is the **grade improvement criteria** for choosing a student).

To summarize, if a student satisfies any one of the three criteria mentioned above, they are chosen as likely to be the best candidates for admission, i.e. they are recommended.

Write a program that asks for a file name, determines the students that are likely to be the best candidates for admission (using the algorithm described above) and prints out their names, scores and the criteria based on which they were chosen. The criteria can be one of “score”, “outlier” or “grade improvement”. If the student isn’t chosen, an empty string is printed for criteria.

If the file cannot be opened, print “`Could not open file.`”

Note: When checking the criteria, check them in the order they are mentioned in the description of the problem i.e. first check for the score criteria, then the outlier criteria and finally the grade improvement criteria. For example, if a student has an Interest value of 0 and their grades improve over the semesters, they will be chosen under the “outlier” criteria and not the “grade improvement” criteria.

Sample File(**superheroes_mini.txt**)

```
Student,SAT,GPA,Interest,High School Quality,Sem1,Sem2,Sem3,Sem4
Abbess Horror ,1300,3.61,10,7,95,86,91,94
Anastasia Kravinoff ,1500,3.74,0,0,92,86,81,90
Adelicia von Krupp ,900,4,5,2,88,92,83,72
Anna Frankenstein ,1050,2.42,5,4,90,91,93,98
Akira Kiamata ,1310,4,1,1,90,86,81,82
Anita Ehren ,1260,2.9,0,0,94,85,82,89
```

Sample Run (using the file above):

```
Enter the file name:
superheroes_mini.txt

Results:
Abbess Horror ,7.7255,score
Anastasia Kravinoff ,5.8045,outlier
Adelicia von Krupp ,5.7875,outlier
Anna Frankenstein ,5.20475,grade improvement
Akira Kiamata , 5.95625
Anita Ehren , 4.6825
```

Explanation:
Abbess Horror has an overall score of ((3.61 * 2) * 0.4) + ((1300 / 160) * 0.3) + (10 * 0.1) + (7 * 0.2) = 7.7255. Since this is greater than 6, they are chosen, i.e. recommended.

Anastasia Kravinoff's overall score is 5.8045 , greater than 5. In addition to this, her interest is 0. She is therefore chosen under the "outlier" criteria.

Adelicia von Krupp's overall score is 5.7875, which is greater than 5. Her normalized GPA is 4*2 = 8 and her normalized SAT score is 900/160 = 5.625. The difference between the two is 8 - 5.625 = 2.375, which is greater than 2. She is therefore chosen under the "outlier" criteria.

Anna Frankenstein's overall score is 5.20475, which is greater than 5. Her interest value is not 0. Her normalized GPA is 2.42*2 = 4.84 and her normalized SAT score is 1050/160 = 6.5625. The difference between the two is 4.84 - 6.5626 = -1.7225, which is not greater than 2. However, her grades improved over the semester. She is therefore chosen under the "grade improvement" criteria.

Akira Kiamata's overall score is 5.95625, which is greater than 5. However, her interest score is not 0 and her normalized GPA (4*2 = 8) and normalized SAT score (1310/160 = 8.1875) do not differ by more than 2. She therefore cannot be chosen under the "outlier" criteria. Since her grades do not improve over the semester, she cannot be chosen under the "grade improvement" criteria either. She is therefore not chosen and so no criteria is printed.

Anita Ehren's overall score is 4.6825, which is less than 5. She is therefore not chosen.

The file should be named `collegeAdmission.cpp`. Don't forget to head over to the code runner on Moodle and paste your solution in the answer box!

Question 4(20pt): Word analysis with arrays
There are several fields in computer science that aim to understand how people use languages. One of the methods is to analyze the most frequently used words by certain writers.

Write a program that reads in a txt file and stores the unique words and their counts in the array and compute the following:

- The top n most frequent words found in the text file
- Total number of unique words
- Total number of words

Program specifications

- Your program should use arrays (not vectors).
- Each line in the text file contains at most 50 words.
- In the txt file, there are at most 200 unique words.
- We have pre-processed the file to remove all punctuation.
- All the words are in lowercase.

- If the file cannot be opened, print "`Could not open the file.`"

Sample File (**miniText.txt**)

```
a true friend will be the true true friend
```

Sample run 1 (**bold** is user input)

```
Enter a filename:
friends.txt
Enter the value of n:
2

3 - true
2 - friend

Unique words: 6
Total words: 9
```

Sample run 2 (**bold** is user input)

```
Enter a filename:
does-not-exist.txt
Could not open the file.
```

The file should be named as `wordAnalysis.cpp`. Don't forget to head over to the code runner on Moodle and paste your solution in the answer box!

Homework 6 checklist

Here is a checklist for submitting the assignment:

1. Complete the **conceptual reviews(mcq)**
2. Complete the code **Homework 6 CodeRunner**
3. Submit one zip file to **Homework 6**. The zip file should be named, **hmwk6_lastname.zip**. It should have the following 4 files:
    - **itemList.cpp**
    - **sells.cpp**
    - **collegeAdmission.cpp**
    - **wordAnalysis.cpp**

Homework 6 points summary

Criteria	**Pts**
Conceptual reviews (MCQ)	10
CodeRunner (problem 1 - 4)	60
C++ file submission (compiles and runs, style and comments)	30
Recitation attendance (March 2, March 3)*	-30
Total	100
5% early submission bonus	+5%
Extra credit: Question 2, average	+5pt

* If your attendance is not recorded, you will lose points. Make sure your attendance is recorded on Moodle.