Homework 6: File I/O

CS16 - Winter 2021

Due:	Thursday, February 18, 2021 (11:59 PM PST)
Points:	80
Name:	Tyler_Pruitt_____
Homework buddy:	_____

- You may collaborate on this homework with **at most** one person, an optional "homework buddy."
- **Submission instructions:** All questions are to be written (either by hand or typed) *in the provided spaces* and turned in as a single PDF on Gradescope. If you submit handwritten solutions write legibly. We reserve the right to give 0 points to answers we cannot read. When you submit your answer on Gradescope, **be sure to select which portions of your answer correspond to which problem** and clearly mark on the page itself which problem you are answering. We reserve the right to give 0 points to submissions that fail to do this.

1. (10 points) When testing for end of file, the lecture (and the readings) mention two methods. What are they and when is each best used?

The first method is to use the '>>' operator in a while loop so the end of the file will be found when this returns false the the loop will stop iterating. This method is best used when the input is numerical data. The second method is to use the '<fstream>' member function '.eof()' which will return a boolean value, often placed inside the conditional of a while loop, if it is at the end of a file. This second method is best for finding the end of the file when the input is treated as text (strings or characters) while using the function '.get()' to read the input from the file.

2. (10 points) Consider the following code snippet from a program where everything is set up correctly:

```
cout << "Enter a line of input: ";
char next;
do {
   cin.get(next);
   cout << next;
} while (!(isdigit(next)) && (next != '\n'));
cout << "END!\n";
```

If the program dialog is as follows:

```
Enter a line of input: I'll see you 2nite at 9PM!
```

then what will be the next line of output? **And explain why.**

```
The next line of output will be

I'll see you 2END!
```

```
This program first prompts the user for input with the cout statement and
initializes the character variable 'next'. Then in the first iteration of
the do-while loop it assigns 'next' to the first character in the input "I",
then prints "I" to screen and then finally checks the while loop condition
to determine if it should keep iterating by making sure that the value of
'next' is neither a digit nor an end line character '\n'. This process
continues for the rest of the sentence including while spaces, letters, and
the apostrophe because all of these characters are neither digits nor an end
line character. On the 14th iteration of this loop, the char variable 'next'
is assigned the value of '2' and then '2' is print out to screen. Then, the
do-while loop condition is checked and it fails since 'isdigit(next)'
returns true. This happens because in this setup of the do-while loop the
variable is first assigned and printed and then checked, not checked then
printed. Finally, the do-while loop ends and the final cout statement prints
"End!" and starts a new line.
```

3. (10 points) I have a text file called “t.txt” that contains two entries: “University of California” on one line, and “Computer Science Rules!” on the next line.

Show the output produced when the following code (entire program not shown so assume all the necessary set ups are done correctly) is executed **and explain why that is**. You are encouraged to also try to compile this to verify your results.

```
ifstream tin;
char c;

tin.open("t.txt");

tin.get(c);
while (!tin.eof()) {
   if ((c != 'e') && (c != 'C')) {
      cout << c;
   }
   tin.get(c);
}
```

The output will be

```
Univrsity of alifornia
omputr Scinc Ruls!
```

First the program initializes the char variable 'c' and ifstream variable 'tin'. Next, 'tin' is connected to "t.txt" and it opens the file to be read. Then, the char variable 'c' is assigned to the first character in "t.txt" which is "C". Then, the while loop condition is checked to see if this is at the end of the file, if it is not the end of the file it prints the char variable 'c' if the value of 'c' is neither 'C' or 'e'. Then, char 'c' is reassigned to the next character is "t.txt". This while loop body executes until the end of the file is reached. The output of this file is the text

```
"University of California
Computer Science Rules!"
```

without the letters "C" or "e", so the real output of this program is the text

```
"Univrsity of alifornia
omputr Scinc Ruls!".
```

4. (10 points) Complete the code below (there are several missing lines) such that the program reads a text file called "MyInputs.txt", which only contains double-type numbers separated by whitespaces. Of course, you don't know ahead of time how many numbers are in the file. You **must** use all the variables declared below, and you may add more variables as needed. Your program should print the average of these numbers, as indicated below.

For example, if the text file contains this single line:

```
4.2 3.3 9.1 3.1 0 0 7.5 5.4 9.9 10
```

Then the program should print out:

```
The average is: 5.25
```

The code is as follows:

```
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
    ifstream inf;
    double num, sum(0), average;
    int count = 0;

    inf.open("MyInputs.txt");

    while (inf >> num)
    {
        sum+=num;
        count++;
    }

    inf.close();

    average = sum/count;

    cout << "The average is: " << average << endl;

    return 0;
}
```

5. (40 points) Write a **function definition** for a function called `FindMedian()` that will read a file that just contains integer numbers that are separated by whitespaces and then finds the *median* value. The median of a set of numbers is the number that has the same number of data elements greater than the number as there are less than the number.

Requirements and Hints:

- You can assume that the input data text file will not have more than 100 numbers in it (but it could have fewer).
- You **have** to write the resulting median (just the integer and newline) in a separate output data file called `median_output.dat`. Don't print anything to standard output.
- The function does not have to check if the input or output files exist prior to reading/writing operations.
- The function definition should only have two arguments: the input filename variable and the `ifstream` variable (see example function call below).
- Hint 1: It is relatively easy to find the median in a set of sorted integers. And, yes, it's OK to define additional functions!
- Hint 2: It matters if the number of integers is odd or even when finding the median. Think about how to address this.

The main function could look like this:

```
int main() {
   ifstream ifs;
   string fname = "inputs.txt"; // could have other names too...
   FindMedian(fname, ifs);
   return 0;
}
```

Submission instructions:

- You will submit this question (just this question) on Gradescope under "HW 06 - Q5" as a C++ file called `median.cpp` (cannot be named otherwise). This should only contain any and all function definitions you have, **except** for `main` (we'll supply that). We will test your submission with an autograder. This has the same due date as this homework.