Name: (as it would appear on official course roster)	
UCSB email address:	**@ucsb.edu**
Lab Section Time:	
Optional: name you wish to be called if different from above	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone")	

Homework 06: File I/O

Assigned: *Tuesday, November 10th, 2020*
Due: *Monday, November 16th, 2020 by 11:59 PM*
Points: *80 (normalized to 100 in grade book)*

- You may collaborate on this homework with AT MOST one person, an optional "homework buddy".
- MAY ONLY BE TURNED IN ON **GRADESCOPE as a PDF file**. Instructions on How to Submit (applicable to ALL homework assignments in this class) are on Piazza.
- There is NO MAKEUP for missed assignments.
- We are strict about enforcing the LATE POLICY for all assignments (see syllabus).
- **IMPORTANT:** If you use code techniques we have NOT covered in class, you will **get a zero grade** on that problem. If you cheat, or have someone else do your work, you will **get an F in the class**.
  Only use the space provided for answers. Use clear and clean handwriting (or typing).

Reading: Chapter 6

ATTENTION: You are turning in TWO things into Gradescope for this homework. The usual PDF as well as a program from Question 5 (see that for details).

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

2. (10 pts) 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**?

3. (10 pts) I have a text file called "**t.txt**" that contains two entries: "University of California" on one line, and "Computer Science Rules!" on the other 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 in a program 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);
}
```

4. (10 pts) 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`

```
#include <iostream>



using namespace std;


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











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





    return 0;
}
```

5. (40 pts) Write a **function definition** 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.
   Givens, 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 + 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 data 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 have additional functions defined!
   - 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"; // file could have other names too…
    FindMedian(fname, ifs);

    return 0;
}
```

   Instructions on how to do this problem:
   You will submit this question (just this question) on Gradescope under "**h06_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 and check your code for style and "class-legal" code, as well as for plagiarism. This has the same due date as this homework.