**CSC 4500            Homework #2 (HW2)         Confirming the Euler Line**

Homework 2 (HW2) is your second CS 4500 assignment, and your first (really small) group assignment. You will be working in a group of 2 or 3 to create this program. Canvas will randomly assign these small groups.

**THE SETUP:** Please watch this Numberphile video about some special points on a Triangle, and how they form an “Euler Line:” <https://www.youtube.com/watch?v=wVH4MS6v23U> I advise you to not attempt writing this program without watching this video from start to finish.

NOTE: The “medicenter” of a triangle is also known as its “centroid” or its “center of mass.”

CAREFUL: there can exist three points that do NOT define a triangle. When? When they are colinear!

Calculating the coordinates of a triangle’s orthocenter: <https://byjus.com/maths/orthocenter/>

Calculating the coordinates of a triangle’s centroid: <https://byjus.com/maths/centroid-of-a-triangle/>

Calculating the coordinates of a triangle’s circumcenter: <https://byjus.com/maths/circumcenter-of-a-triangle/>

The circumcenter is the most difficult of the three to calculate. One thing to keep in mind is that the three perpendicular bisectors of the triangle’s sides meet at the same point. Therefore, you only need find where two of the bisectors meet – the third is guaranteed to meet at that same point. (<https://www.merriam-webster.com/dictionary/circumcenter>)

**THE ASSIGNMENT:** Your program should start by printing out to the screen a short but understandable description of what the program will do. Then the program should prompt the interactive user for three NON-COLINEAR points in Cartesian two dimensional space. For each point, prompt the user for an X coordinate and a Y coordinate. For each coordinate, the user may type in an integer, or a real number using a decimal. The numbers can be positive, negative, or 0. If the user types in something inappropriate (like a non-number), the program should give a descriptive error message, and reprompt.

After the program has three points, the program should check to make sure that the three points are not colinear. If the three points ARE colinear, then the program should print out an error message to that effect, pause until the user presses ENTER, and then halt.

If the three points are NOT colinear, then your program should calculate the resulting triangle’s orthocenter, and print out its coordinates; should calculate the resulting triangle’s centroid, and print out its coordinates; and should calculate the resulting triangle’s circumcenter, and print it out. It should be clear to the interactive user what these coordinates are; that is, label your output in an understandable way.

The three points you calculate should be colinear, except for one special case: an equilateral triangle. If the triangle entered is an equilateral triangle, then the three “centers” should all be the same point. If the user entered an equilateral triangle, your program should print out a message to that effect, and also print out a message telling the interactive user the absolute value of the biggest distance between the three calculated centers of that triangle. HINT: That distance should be small. Then the program should prompt for an ENTER from the user; when the user pushes ENTER, the program should halt.

If the three points are not colinear, and unless the triangle entered is an equilateral triangle, then the three centers should be separate, but colinear points. Pick two of the three centers that resulted from the triangle coordinate that were input by the users, and calculate the equation of the line that passes through those two centers. (I don’t care which two centers you pick, but you should output data so that the interactive user knows which two centers you picked.) Measure the closest distance from the third center to that line; call that value D. HINT: Because of the Euler line, D should be small, or 0. The Euler line hypothesis suggests it should be zero, but round off errors in your calculations may make it non-zero. When you print out D, include D, and also include the percentage error, dividing D by the length of the line segment between the two points your program used to determine the Euler line. Round that percentage to two decimal points. For example, if the distance between the two points is 11.3, and the distance D is 0.14, then the percentage error is 100\* (0.14/11.3)%, or approximately 1.24%.

Once the program has output the required information as described above, it should prompt for ENTER from the interactive user. When the user pushes ENTER, the program should halt.

**EXTRA FOR EXPERTS:** Just for fun, it would be cool to graph the triangle and its three centers. This is NOT required, but if you want a challenge, go for it!

**WHICH PROGRAMMING LANGUAGE MUST YOU USE?** As before, you can use any of the programming languages on our “approved list:” C, C++, Java, JavaScript, Pascal, and Python. However, you and your partner (or partners) must make a group decision about what language you should use. Your group will submit only one program, so you should not produce multiple programs for us to grade. As before, no matter what language you choose, you must use the appropriate IDE in this set of online compilers: [https://www.onlinegdb.com](%20https://www.onlinegdb.com%20)  What you will submit (to Canvas) is your source code as a text file. We will copy your code, paste it into OnlineGDB, and run it. If it works, great. If it does not work (first time), then you and I have a problem, and that problem will cost me time and it will cost you points. It is not my job to debug your submitted program. However, I am more than willing to help you before the assignment is due.

**PLEASE NOTE WELL:** I want a text file submitted. NOT an MS Word file, not an OnlineGDB project file, not a PDF. *Text file.* I emphasize this because in the past far too many people ignored this direction. They lost points, and I lost time.

**PROGRAMMING TIPS:** It is better to hand in something simple that works, and on time, rather than handing in something fancy that is either late or doesn’t work. The best HW2 programs will work, be on time, be well documented (see below), and will do something interesting.

**WHAT IS GOOD DOCUMENTATION?** The specification above mentions “well documented.” All documentation must be in the English language; spelling and grammar count. Here are some hints of what good documentation might look like:

**Start your program with an “opening comment.”** You may include more items than the items listed here, but you may NOT leave out any of the items mentioned here. If you DO leave any of these out, or if you do a poor job on any of them, you will lose points. If one of these items is not applicable to your program (for example, if it doesn’t use any external files), then include the name of the item in your opening comment but add “N/A” or “none” to indicate that this isn’t applicable to your program.

**Items to include in your opening comment:**

* a title for your program
* the name of the file that holds your program
* the name of the programming language you are using, and the IDE to use
* a list of any external files necessary to run your program
* a list of external files your program creates (If you list any external files, briefly explain what each of them contains.)
* the names of any programmers working on the program (for HW1, this is just you)
* email address of all programmers
* the course number and section number of the course you’re writing this program for
* the date you finished the program and submitted it
* an explanation of what the program does
* any resources you used to complete the program (Always give credit where credit is due; for example, if you used a website to check on an algorithm, list that here.)

**Other comments that are required:**

* each declaration should include a comment that explains its use
* each subprogram (function, subroutine, object,…) should have an opening comment describing its purpose)
* each loop should be preceded by a comment that describes its purpose
* any statement that is particularly unclear or “tricky” should have a comment clarifying its use to the human reader

**How I will grade submitted programs (total, 100 points). Although this will be fairly consistent during CS 4500, I may deviate from some of the details for particular programs:**

* 10 pts: Was the program handed in on time in a form consistent with the instructions?
* 10 points: Did your program compile without errors using the specified IDE?
* 30 points: Is the program well documented (see above)?
* 40 points: Does the program work EXACTLY as specified? Includes appropriate messages to the user.
* 10 points: Is the program reasonably efficient?