Lab #4 – Programming Assignment
Due: Friday February 26, 2021 at 11:00pm (time local to West Lafayette, IN)
10 Points Possible

Problem: Given three points that will always create a triangle with a positive area, calculate the distance between the points and the coordinate of the incenter found within the triangle created. For this assignment you will be **required** to implement the user-defined functions (from chapter 4).

Example Execution #1:

```
Enter X coordinate #1 -> 13
Enter Y coordinate #1 -> 20
Enter X coordinate #2 -> 25
Enter Y coordinate #2 -> 40
Enter X coordinate #3 -> 30
Enter Y coordinate #3 -> 15

Distance from point 1 to point 2: 23.32
Distance from point 2 to point 3: 25.50
Distance from point 3 to point 1: 17.72
Location of incenter: 22.15, 23.57
```

Example Execution #2:

```
Enter X coordinate #1 -> -5
Enter Y coordinate #1 -> 25
Enter X coordinate #2 -> 15
Enter Y coordinate #2 -> 0
Enter X coordinate #3 -> 10
Enter Y coordinate #3 -> 30

Distance from point 1 to point 2: 32.02
Distance from point 2 to point 3: 30.41
Distance from point 3 to point 1: 15.81
Location of incenter: 5.18, 21.99
```

Example Execution #3:

```
Enter X coordinate #1 -> 0
Enter Y coordinate #1 -> -5
Enter X coordinate #2 -> 35
Enter Y coordinate #2 -> 25
Enter X coordinate #3 -> 30
Enter Y coordinate #3 -> -10

Distance from point 1 to point 2: 46.10
Distance from point 2 to point 3: 35.36
Distance from point 3 to point 1: 30.41
Location of incenter: 21.88, 1.10
```

Example Execution #4:

```
Enter X coordinate #1 -> 55
Enter Y coordinate #1 -> -5
Enter X coordinate #2 -> 4
Enter Y coordinate #2 -> -2
Enter X coordinate #3 -> 30
Enter Y coordinate #3 -> -10

Distance from point 1 to point 2: 51.09
Distance from point 2 to point 3: 27.20
Distance from point 3 to point 1: 25.50
Location of incenter: 30.17, -6.72
```

Example Execution #5:

```
Enter X coordinate #1 -> 5
Enter Y coordinate #1 -> 5
Enter X coordinate #2 -> -5
Enter Y coordinate #2 -> 0
Enter X coordinate #3 -> -2
Enter Y coordinate #3 -> -8

Distance from point 1 to point 2: 11.18
Distance from point 2 to point 3: 8.54
Distance from point 3 to point 1: 14.76
Location of incenter: -1.55, -1.35
```

All course programming and documentation standards are in effect for this and each assignment this semester. Please review this document!

Additional Requirements:

1. Add the **lab assignment header** (`vi` shortcut `hlb` while in command mode) to the top of your program. An appropriate description of your program must be included in the assignment header. Include the Purdue University e-mail addresses of **each contributing group member** in the assignment header!
2. **Each of the example executions provided for your reference represents a single execution of the program.** Your program must accept input and produce output **exactly** as demonstrated in the example executions, do not add any "bonus" features not demonstrated in the example executions. Your program will be tested with the data seen in the example executions and an unknown number of additional tests making use of **reasonable data.**
   - All floating-point variables must be of the `double` type.
   - All input will be of the integer (`int`) type.

Additional Requirements (continued):

3. For this assignment you will be **required** to implement the user-defined functions (from chapter 4). Failing to follow course standards as they relate to good user-defined function use will result in a **zero for this assignment.**
4. Revisit **course standards as it relates what makes for good use of user-defined functions, what is acceptable to retain in the `main` function, and when passing parameters by address is appropriate.**
   - In many cases user-defined function use should result in a `main` function that only declares variables and makes calls functions.
5. Course standards **prohibit** the use of programming concepts not yet introduced in lecture. For this assignment you can consider all material in the first four chapters of the book, notes, and lectures to be acceptable for use.
6. A program **MUST** compile, be submitted through the `guru` server prior to the posted due date to be considered for partial credit. The C-file you submit must be named exactly: `lab04.c`

Course Programming and Documentation Standards Reminders:

- Use the course function header (`head_fx vi` shortcut `hfx` while in command mode) for every user-defined function in your program.
  - List and comment **all parameters** to a function, one per line, in the course function header.
  - **All function declarations** will appear in the global declaration section of your program.
  - **The user-defined function definitions will appear in your program after the `main` function.**
- Maximize your use of symbolic/defined constants and minimize your use of literal constants.
- Indent all code found within the `main` and all user-defined functions **exactly** two spaces.
- Place a **single space** between all operators and operands.
- Comment **all** variables to the right of each declaration. Declare only one variable per line.
- Notice that several programs (see program 2-9 on pages 74-75) in the programming text use a single line comment to indicate the start of the local declaration and executable statement sections of the `main` function.
  - At no point during the semester should these two sections ever overlap. You might consider adopting this habit of commenting the start of each section to help you avoid this mistake.
- Select **meaningful identifiers** (names) for all variables in your program.
- Do not single (or double) space the entire program, **use blank lines when appropriate**.
- There is no need to include example output with your submission.

Auto-Grade Tool

- We have implemented what is being referred to as the auto-grade tool. At the time of a successful assignment submission you may receive some feedback on your program in regards to course programming and documentation standards. This feedback may include a potential deduction that you will receive once your assignment is reviewed by your grader.
- It is expected that graders verify those notes identified by this tool to ensure that they are indeed applicable and reasonable to the submission. Graders may make additional deductions for those standards not identified by the new tool.
- We hope that this feedback helps with the enforcement of course standards, consistency in grading across sections, and to encourage students to revise their work when problems are identified before the assignment deadline passes. It is possible to resubmit an assignment for grading up to the advertised deadline. Only the final successful submission is retained and evaluated.