**Walking in Medford Lakes** *(this assignment is designed for MySQL Spatial)*

1. Using the diagram of Medford Lakes ([http://jackmyers.info/db/exercises/grad/medfordlakes/medfordlakes.pdf](http://jackmyers.info/db/exercises/grad/medfordlakes.pdf)),   
   create and populate the following tables in MySQL  
   * **POI** (point of interest): store a primary key, the name of the POI, and the X, Y location
     + PJ Whelihans
     + Medford Lakes Country Clubhouse
     + the Zinc Café
     + YMCA Camp Ockanickon
     + 13 Big Chief Trail  
       *(at the intersection of Big Chief Trail and Big Look Trail)*
     + Chicagami Trailhead (80, 21)
     + Upper Aetna Lake Dam (90, 31)
     + Medford Lakes private island
   * **Lake**: store a primary key, the name of the lake, and the lake’s geometry
     + Lower Aetna Lake
     + Upper Aetna Lake (don’t forget to account for the private island in the lake)
   * **Person:** store a primary key, then name of the person and the person’s X, Y location. You can make as many people as you like.
2. Create a person for you. Your mission is to start walking/swimming from one POI to another in a straight line by calling a procedure named move with inputs of a person id, and two poi\_ids. You will place your person at the starting location then “step through” X and Y coordinates on your journey until the person is at the destination. However, you cannot move more than one unit in either the X or Y direction. (You’re not Wonder Woman or Superman.) Only integer coordinates allowed. If your journey takes you into a lake, indicate that.  
     
   HINTS:
   * Your MySQL procedure needs to calculate the slope of the line between POIs and its y intercept.
   * You need to know if you are predominantly moving northward, southward, eastward, or westward to properly move. Eastward/Westward movements increment X and calculate Y. Northward/Southward movements increment Y and calculate X.
   * MySQL procedures do not let you write to the console directly. A work-around technique is to create a table called procedure\_log used to store messages with an autoincremented id and a varchar message field. Then you could build a helper procedure to log messages.

CREATE PROCEDURE `log`(IN line varchar(255))

BEGIN

INSERT into procedure\_log (message) VALUE (line);

END

You could use that helper procedure in your “walking” procedure as follows:

TRUNCATE procedure\_log; -- to clean out the log from its last use.

CALL log('Let''s take a walk in Medford Lakes');

CALL log('');

And when ready to display the messages:

SELECT message FROM procedure\_log ORDER BY message\_id;

1. What to turn in?  
   - A copy of all your code that creates and populates tables.  
   - In the same file, a copy of the code that creates your procedure.  
   - Screen shots of your journeys (please call the procedure five times for five journeys:
   * One from the Upper Aetna Lake Dam to the Chicagami Trailhead (as shown above);
   * One essentially heading north;
   * One essentially heading south;
   * One essentially heading east;
   * One essentially heading west.

Here’s a sample run from my solution program

Text, table

Description automatically generated