CS585 Summer22 - HW3 Rubrics (6 points)

1. (1 point) KML File

kml file from step 5 - with the placemarks, convex hull and nearest-neighbor line segments

- First, use one of the following websites/tools:
  - https://kmlviewer.nsspot.net/
  - https://www.doogal.co.uk/KmlViewer
  - Go to Google Earth → Click the menu bar on the right → Click Projects → Click Open → Select “Import KML file from computer”
- Check the following
  i. -0.25 each for missing locations. There should be 13 location pins.
  ii. -0.5 for incorrect convex hull (all points/pins should be within the convex hull and a few should be on the boundary)
  iii. -0.5 for incorrect nearest neighbor (there should be a line from one point – labeled as “home” location – to the other point. Visually verify that there isn’t be any other point that is closer from their “home” point than the one with a line)

Note: You may need to test the file on multiple websites/tools because the nearest neighbor line may not show up properly.

2. (no points, but ..) Pictures of Locations

a. There should be 13 images.
b. -2 points if the student submits less than 6 images
c. -1 point if the student submits more than 6 but less than 10 images
d. Okay to miss at most 3 images (no deduction)

3. (2 points) SQL Commands

A text file(s) (.txt or .sql) with your two queries (a file for each query is okay)

Computing Convex Hull (1 point)

- -1 point if no convex hull command, but only table creation and data insertion commands. Check to see if there’s a query for computing the convex_hull (e.g., *ST_CONVEXHULL* for POSTGRES)
- If you are not sure about the correctness of the query, please run the query

Computing the nearest neighbor (1 point)

- -1 point if no nearest neighbor command, but only table creation and data insertion commands. Check to see if there's query for getting a nearest neighbor (e.g., *ST_Distance* for POSTGRES)
- -0.5 points if their query doesn't have some sort of ordering/sorting command in it.
- If you are not sure about the correctness of the query, please run the query.

Note:

- Okay if they hardcode points
- Okay if they create and use a table to store the 13 points and write queries for the table.
- Okay if the nearest neighbor command outputs the entire table row.

4. (1 point) Screenshot and JS OpenLayers Code (CodePen/jsFiddle - okay)

- -1 point if neither is submitted

The screenshot (0.5 points)

- 0.5 points if no screenshot of Google Earth with all 13 points in it (similar to 1) - the screenshot should show all points are on/inside the convex hull and show the nearest neighbor line.

The code (0.5 points)

- If they submit the html file, make sure that you can run the file using either: https://bytes.usc.edu/~saty/tools/xem/run.html?x=OpenLayers or https://bytes.usc.edu/~saty/tools/xem/run.html?x=OpenLayers_v2
- If they give a link to CodePen/jsFiddle, follow the link and make sure that you can run the code.
- -0.25 points for each missing point in the map. You may need to zoom out to see all the points (count them)
- -0.5 points for not using localStorage (if the points directly get plotted, without being stored and retrieved). Check to see if you can something like the following

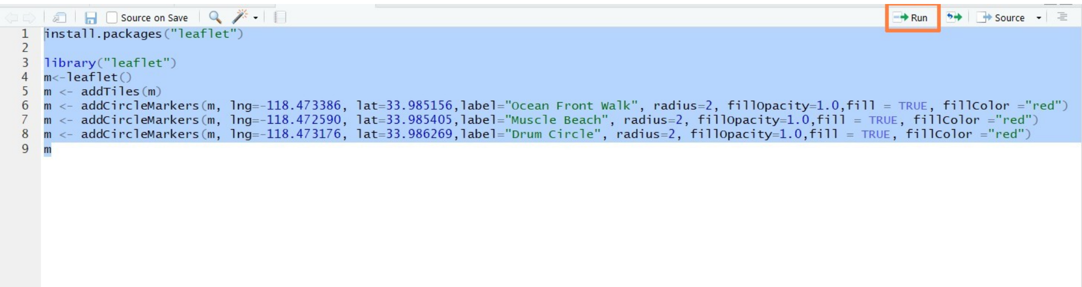
  localStorage.setItem(xxx, thedata); //– store the data
  Somevar = JSON.parse(localStorage.getItem(xxx)) //parse/retrieve the data
  Someothervar = somevar.points //use the data

5. (1 point) Visualization and R script

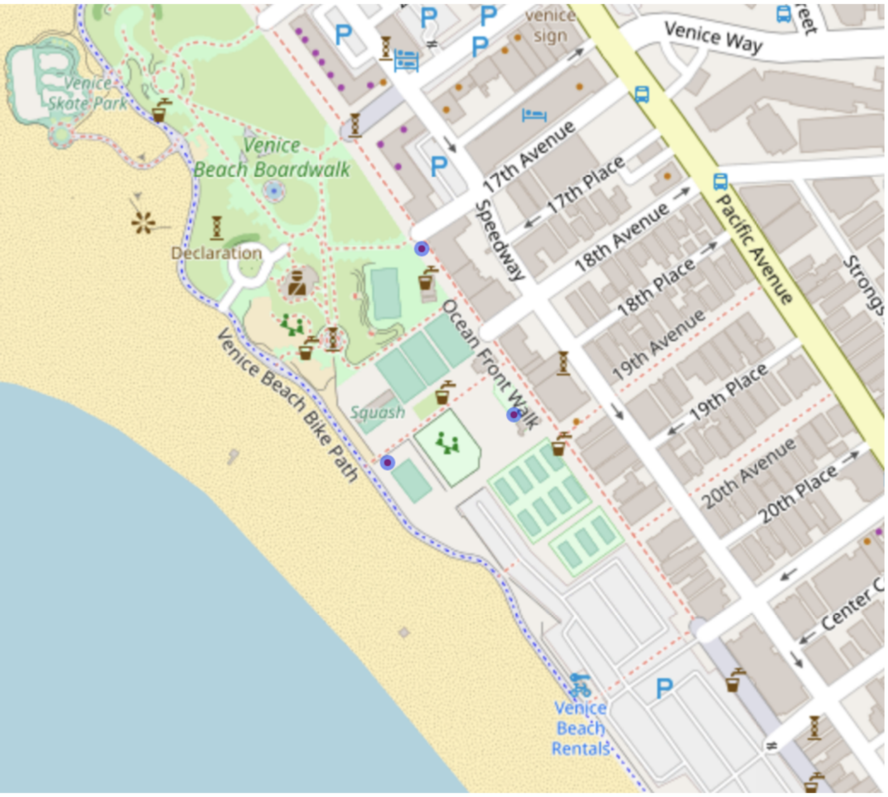
from step 7: a screenshot of the visualized locations, plus your .R script

a. -0.5 points for incorrect or missing the location screenshot (e.g. wrong output points).
b. -0.5 points for incorrect or missing the R script (e.g. wrong parameters or the code doesn't run).

Sample script and output:



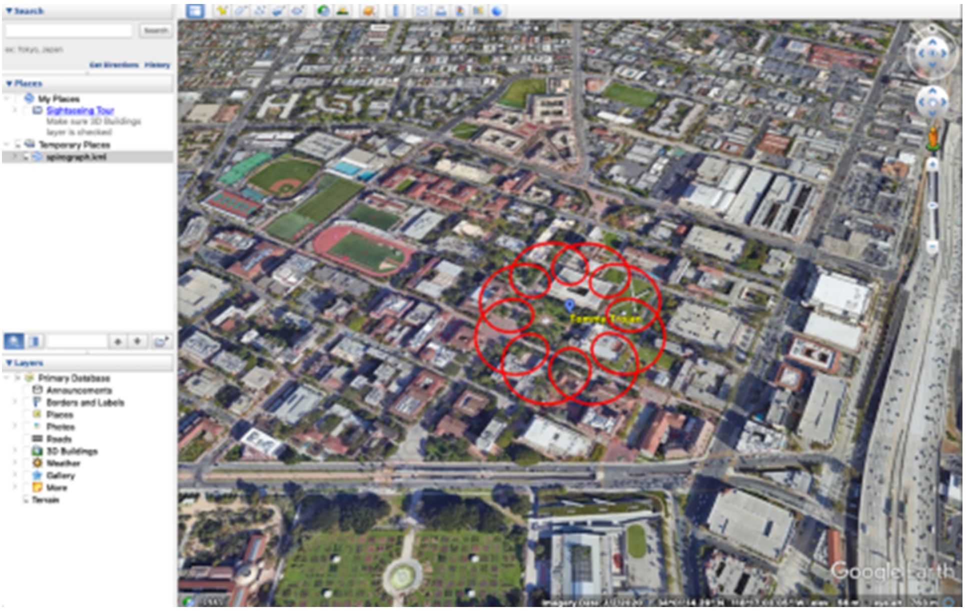
```
install.packages("leaflet")

library("leaflet")
m<-leaflet()
m <- addTiles(m)
m <- addCircleMarkers(m, lng=-118.473386, lat=33.985156,label="Ocean Front Walk", radius=2, fillOpacity=1.0,fill = TRUE, fillColor ="red")
m <- addCircleMarkers(m, lng=-118.472590, lat=33.985405,label="Muscle Beach", radius=2, fillOpacity=1.0,fill = TRUE, fillColor ="red")
m <- addCircleMarkers(m, lng=-118.473176, lat=33.986269,label="Drum Circle", radius=2, fillOpacity=1.0,fill = TRUE, fillColor ="red")
m
```



6. (1 point) Spirograph

a screenshot of your Spirograph™ result from step 8, plus the source code in text form (eg. spiro.{js,py,java,cpp}), .kml, .shp.

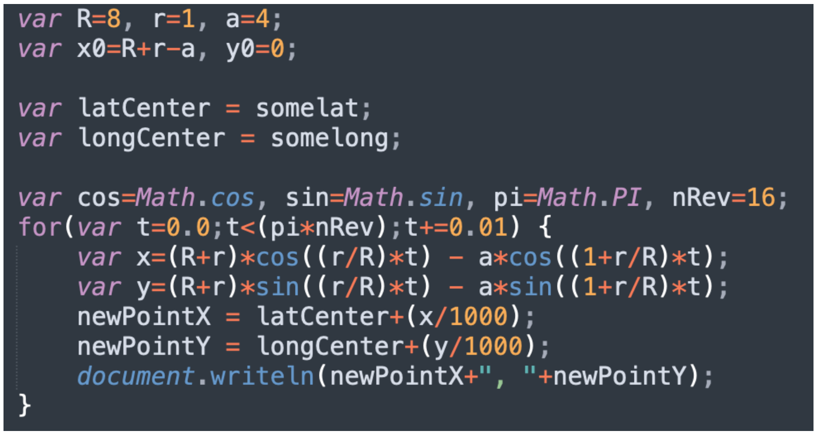
a. -0.25 for missing or incorrect screenshot. (eg. The center for the spiral should be SGM123 as the center for on campus students. For off campus/DEN students, the center of the spirograph can be their home) Note that the spirograph should **contain 8 loops.**



b. -0.25 for missing or incorrect source code (js, py, java, cpp, etc). (e.g. the code doesn’t run or the logic/calculation is incorrect – pay close attention to the x and y calculation equations as shown in the sample pic below).

(R = 8, r = 1, a = 4)

Sample:



```
var R=8, r=1, a=4;
var x0=R+r-a, y0=0;

var latCenter = somelat;
var longCenter = somelong;

var cos=Math.cos, sin=Math.sin, pi=Math.PI, nRev=16;
for(var t=0.0;t<(pi*nRev);t+=0.01) {
    var x=(R+r)*cos((r/R)*t) - a*cos((1+r/R)*t);
    var y=(R+r)*sin((r/R)*t) - a*sin((1+r/R)*t);
    newPointX = latCenter+(x/1000);
    newPointY = longCenter+(y/1000);
    document.writeln(newPointX+", "+newPointY);
}
```

c. -0.25 for missing or incorrect .kml file (e.g. wrong format, or wrong values that can be recognized from the output spirograph). Again you can use the services/websites/tools (in 1) to run their KML file to verify. It should output the spirograph.

d. -0.25 for missing a shape file (there should be a .zip file in their submission - You don’t need to unzip it. If a zip file is present, no deduction. But feel free to unzip to verify. There should be files like .shp, .dbf, or .shx extensions in it.)