Part 1: KotlinAssignment.kt

In this part of the assignment you will be practicing with Kotlin variables, functions, and control structures. Complete this part by implementing the described functions. You can and should **call** these functions from inside the `main()` method in order to test your work.

Your program should do all the following, pay attention when it says you must develop a function to do something:

1. Step 1: Define a _data class_ called `City` that will represent the location of a city. A City should have three properties: a `name` (String), `latitude` (Double), and `longitude` (Double).

2. Step 2: Define **a function** called `listCities()` that returns a `list` of City objects representing the following U.S. cities:
   {name: "Atlanta", latitude: 33.7490, longitude: -84.3880}
   {name: "Bozeman", latitude: 45.6770, longitude: -111.0429},
   {name: "New York", latitude: 40.7128, longitude: -74.0060}
   {name: "San Francisco", latitude: 37.7749, longitude: -122.4194},
   {name: "Tacoma", latitude: 47.2529, longitude: -122.4443},
   Call your function from inside the `main()` method, saving the result to a local variable called `cityLocations`. Print this out to check your work.

3. Step 3: Define a function `distanceFromSeattle()` that takes as a param a City object and return the distance in kilometers (Double) between that city and Seattle, WA (latitude: 47.6062, longitude: -122.3321).
   You can use the `haversine()` function provided below to calculate the distance in km between two points on the globe.
   Call your function from inside the `main()` method and print out the distance between Seattle and Tacoma (should be about 40km)

```
//The following is the haversine function you should incorporate to find the
distances.

//Calculates the distance in km between two points on a globe

//Implementation from http://rosettacode.org/wiki/Haversine_formula


fun haversine(lat1: Double, lon1: Double, lat2: Double, lon2: Double): Double {
    val R = 6372.8 // in kilometers
    val l1 = Math.toRadians(lat1)
    val l2 = Math.toRadians(lat2)
    val dl = Math.toRadians(lat2 - lat1)
    val dr = Math.toRadians(lon2 - lon1)
    return 2 * R * Math.asin(Math.sqrt(Math.pow(Math.sin(dl / 2), 2.0) +
           Math.pow(Math.sin(dr / 2), 2.0) * Math.cos(l1) * Math.cos(l2)))
}
```

4. Step 4: From your `main()` method, call the `map()` method of the `cityLocations` list, passing it your `distanceFromSeattle()` function in order to get a list of distance between the cities and Seattle. Print the resulting list.

5. Step 5: From the `main()` method, call the `filter()` method of the `cityLocations` list, passing it an _anonymous function_ used to filter the list. This function should return `true` if the given City is "far" from Seattle (further away than 1000km). Print out the resulting list.

6. Step 6: From the `main()` method, use the `map()` and `filter()
   methods to get a list of the `names` of cities that are in the "West",
   meaning west of the Mississippi river. For our purposes, the Mississippi is
   at longitude -89.97803 E (so anything "less than" that is to its west).
   Use two _lambda expressions_ to do this; utilize the implicit `it` parameter.
   Print out the resulting list.
   Hint: filter for western cities, and then map the results to a list of names
   (Strings).

7. Step 7: From the `main()` method, use the `maxBy()` method of the
   `cityLocations` list to find which city is the furthest from Seattle.
   Since the last (and only) parameter to the `maxBy()` method is a function,
   you should omit the parentheses from the method call.
   Print the resulting city.
   Hint: https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.collections/max-by.html

8. Step 8: From the `main()` method, use the map method to produce a
variable called `mappedCities` that is a `Map<String, City>` object, where
each key is the name of a city (String) and the value is the City object.
Print the resulting map, one line per key, value object.

Then use mappedCities to print the latitude of Bozeman by using the key value to identify the proper value. Make sure you put your print statement in an if statement that checks to see if “Bozeman” is present in mappedCities.

Here is the output of Steps two through 8 (I misspelled San Francisco…..on purpose):

You should mimic your output to look as similar to this as possible……but spell San Fran correctly

```
**********Step Two**********
City(name=Atlanta, latitude=33.749, longitude=-84.388)
City(name=Bozeman, latitude=45.677, longitude=-111.0429)
City(name=New York, latitude=40.7128, longitude=-74.006)
City(name=San Fransico, latitude=37.7749, longitude=-
122.4194)
City(name=Tacoma, latitude=47.2529, longitude=-122.4443)
**********Step Three**********
The distance between Seattle and Tacoma is
40.19290553996769 km
**********Step Four**********
A new List with the distances of all cities in list to
Seattle:
[3506.672723087685, 887.4977864782966,
3866.6204789528556, 1093.5226297469185,
40.19290553996769]
**********Step Five**********
List of cities over 1000KM from Seattle (used the return
list from the Filter function and just printed the city
name)
Atlanta
New York
San Fransico
**********Step Six**********
[Bozeman, San Fransico, Tacoma]
**********Step Seven**********
```

```
Max Distance: City(name=New York, latitude=40.7128,
longitude=-74.006)
**********Step Eight**********
Name Atlanta: City City(name=Atlanta, latitude=33.749,
longitude=-84.388)
Name Bozeman: City City(name=Bozeman, latitude=45.677,
longitude=-111.0429)
Name New York: City City(name=New York,
latitude=40.7128, longitude=-74.006)
Name San Fransico: City City(name=San Fransico,
latitude=37.7749, longitude=-122.4194)
Name Tacoma: City City(name=Tacoma, latitude=47.2529,
longitude=-122.4443)
The latitude for Bozeman is 45.677
```