

**Programming Exam 1**

This test has 1 question. You have 50 minutes. The exam is open book, open note, and open web. You may use code from your programming assignments or the Introduction to Programming in Java booksite. No communication with any non-staff members is permitted. Submit your solution via Dropbox. **Write out and sign the Honor Code pledge before turning in the test.**

*"I pledge my honor that I have not violated the Honor Code during this examination."*

-----  
Signature

**Name:**

**NetID:**

**Precept:**

**Do not remove this exam from the exam room.**

**Problem.** You are visiting a large city in a foreign country, and you want to find a restaurant for lunch. You skipped breakfast, so you are very hungry. You have your smartphone with you, but you forgot to get international roaming enabled, so you have no network service. However, before you left home, you used Google Maps to download the (latitude, longitude) coordinates of several of the restaurants in the city, as well as their names and average user ratings, and put them in a text file on your smartphone's SD card. Fortunately for you, your smartphone can run Java programs, and its GPS receiver is working and can give you your current (latitude, longitude) coordinates.

Your task is to quickly write a Java program `FindFood.java` to help you find a place to eat. It should take three command-line arguments: your latitude and longitude in degrees (both `double`), and your minimum acceptable restaurant rating (an `int` between 1 and 5, inclusive). It will read the text file with the restaurant information via `StdIn`, and print out the restaurant names, distances to you, and ratings, **in order from shortest distance to farthest distance**. You must complete this program within 50 minutes, before crippling hunger prevents you from writing workable Java code.

**API Specification** Your program `FindFood.java` must be organized as a library of public static methods with the following API:

```
public class FindFood
-----
double dist(double x1, double y1,           // Calculate the spherical distance between
           double x2, double y2)          // (x1, y1) and (x2, y2). Compute the distance
                                           // using the Great Circle formula for land miles.

int findNext(double x, double y,           // Given my latitude and longitude as (x, y) and
           double[] lat, double[] lon)    // the latitudes and longitudes of the restaurants
                                           // as (lat[i], lon[i]),
                                           // return the integer index i of the restaurant
                                           // closest to me.

void markFound(int f, double[] lat,        // Given an index f that identifies a restaurant
               double[] lon)             // and the latitudes and longitudes of all the
                                           // restaurants, change the latitude and longitude
                                           // of restaurant f to Double.POSITIVE_INFINITY
                                           // so that f will be ignored by findNext()

void main(String[] args)                  // 1. Input user position and minimum rating from
                                           // the command-line.
                                           // 2. Read each restaurant's latitude, longitude,
                                           // name, and rating from standard input and
                                           // store in parallel arrays.
                                           // 3. Find the closest restaurant and, if it has
                                           // an acceptable rating, print its name, distance
                                           // and rating. Mark it as found, and repeat 3
                                           // until all acceptable restaurants are printed.
```

Because we are using (latitude, longitude) coordinates, you should NOT use Euclidean distances. Instead, use the **Great Circle distance formula** for statute (land) miles:

$$distance = 1.1516 \times 60 \arccos(\sin x_1 \sin x_2 + \cos x_1 \cos x_2 \cos(y_1 - y_2))$$

where  $(x_1, y_1)$  and  $(x_2, y_2)$  are the coordinates (latitude, longitude) in degrees of the two points. This differs from the Great Circle distance formula you used in your first assignment only by a factor of 1.1516 which converts nautical miles to statute (land) miles. Remember to convert the latitudes and longitudes to radians using `Math.toRadians()` before using the Math library's trigonometric functions, and be sure to convert the result back into degrees using `Math.toDegrees()`. The Java Math function for arccos is `Math.acos()`.

Warning: Do not use `long` as a variable name for longitude. `long` is a reserved word in Java.

**Data.** The restaurant data file has the number of restaurants on the first line. Each subsequent line has the name, latitude, longitude, and rating for one restaurant. All ratings are between 1 and 5, inclusive. Restaurant names do not have spaces. For example:

```
% more 5restaurants.txt
5
Delmonico's          40.708881 -74.009943 4
TheRussianTeaRoom   40.768712 -73.980074 3
GreenwichVillageBistro 40.730348 -74.00239 5
PerSeRestaurant     40.731259 -74.00239 2
TheCorneliaStreetCafe 40.730478 -74.00239 5
```

The following data files are available for testing your program:

```
http://introcs.cs.princeton.edu/data/5restaurants.txt
http://introcs.cs.princeton.edu/data/10restaurants.txt
http://introcs.cs.princeton.edu/data/15restaurants.txt
```

**Sample runs.** Suppose I'm at (40.75, -73.98), and I want to find restaurants near me with ratings of at least 3. Then, my program should behave as follows:

```
% java FindFood 40.75 -73.98 3 < 5restaurants.txt
TheRussianTeaRoom has a rating of 3
Distance: 1.292930 miles

TheCorneliaStreetCafe has a rating of 5
Distance: 1.787035 miles

GreenwichVillageBistro has a rating of 5
Distance: 1.793825 miles

Delmonico's has a rating of 4
Distance: 3.245044 miles
```

Your program should work for other input as well.

```
% java FindFood 45.49 -73.59 5 < 10restaurants.txt
Toque has a rating of 5
Distance: 1.650935 miles
```

**Print Specification** We leave it for you to write the print statement for the first line of output for each restaurant, but for the second line, use the following print statement to get the same number of decimal places and separating lines as the sample run:

```
StdOut.printf("Distance: %.6f miles\n\n", d);
```

Of course, your variable names may differ from ours.

**Submission.** Submit the single file `FindFood.java` via Dropbox at

[https://dropbox.cs.princeton.edu/COS126\\_S2012/Exam1](https://dropbox.cs.princeton.edu/COS126_S2012/Exam1)

Be sure to click the *Check All Submitted Files* button to verify your submission.

**Grading.** *Your program will be graded on correctness and clarity. You will receive partial credit for correctly implementing the following components:*

- *Header with name, login, precept*
- *Reading the input data and storing it in four parallel arrays.*
- *Printing appropriately formatted output.*
- *The `dist()` function.*
- *The `findNext()` function.*
- *The `markFound()` function.*

*You will receive a substantial penalty if your program does not compile or if you do not follow the prescribed API or input/output specifications.*

*For partial credit: If you are stumped on step 3 of `main()` then just print out all the restaurants and their ratings and distances to your location in the same format as the sample output.*