Please solve the 2 following problems, they have separate deliverables. You can perform this in any programming language.

**Problem 1:**  
  
Create a REST API for a system that assigns students to classes. API can be used by both a UI and programmatically by other systems.  
  
**Deliverables:**  
Deliver the following to

[set recipients]

(required) code via a GitHub repository.  
(required) a short write-up around what technologies/frameworks you are/would use in implementing various parts/tiers of this system  
(optional) deployable/runnable artifact  
(optional) API documentation  
  
**Timeframe:**  
The scope of the exercise is somewhat fluid so do not spend more than 8 hours on it.   
  
**Detailed Requirements:**  
  
Models:  
Student = { student id, last name, first name }  
Class = { code, title, description }  
Student can attend an unlimited number of classes. Classes can have an unlimited number of students  
  
Operations:  
Create/Edit/Delete Student  
Create/Edit/Delete Class  
Browse list of all Student  
Browse list of all Classes  
View all Students assigned to a Class  
View all Classes assigned to a Student  
Search Student/Classes by available fields/associations  
  
**Security:**  
None  
  
**Error Handling:**  
Does not need to be thorough. Just enough to demonstrate how you would handle various types of errors (business, system)  
  
**Persistence:**  
Not part of the evaluation. Feel free to mock it if that’s faster.

**Problem 2:**  
  
Given each student has a geolocation lat/lon point, how would you determine which students are physically in any classroom?   
  
Write a function that returns the students if they are in a classroom.   
  
  
**Assumptions**  
   
Each classroom has a square shape of 20m X 20m and none of the classrooms intersect.  
Students are dimensionless outside of their latitude / longitude point  
Height is not a concern for either the student or the classroom  
It doesn’t matter which student was in which classroom, we only care about the list of students found  
This is intended to be performed in memory where you don’t have the usage of a database of some sort.   
  
NOTE: If you made any other assumptions please write those down.  
  
  
  
**Function Description**  
  
Types:  
  
Student = { name , latitude, longitude }  
Classroom: { name, latitude, longitude } (the latitude and longitude represent the exact center of the classroom, which we assume to be a 20m x 20m square)  
StudentList = list of Students (java: List<Student>, python: [])  
StudentFoundList = list of Students found (java: List<Student>, python: [])  
ClassroomList = list of Classrooms (java: List<Classroom>, python: [])  
  
Function Signature:  
  
StudentFoundList = studentsInClasses(StudentList, ClassroomList)  
  
  
  
  
  
  
**Expected Input/Outputs:**  
  
Example 1:  
  
Input Variables:  
  
engineering\_classroom = { 'name': 'Principles of computational geo-location analysis', 'latitude': 34.069140, 'longitude': -118.442689 }  
geology\_classroom = { 'name': 'Sedimentary Petrology', 'latitude': 34.069585, 'longitude': -118.441878 }  
psychology\_classroom = { 'name': 'Introductory Psychobiology', 'latitude': 34.069742, 'longitude': -118.441312 }  
music\_classroom = { 'name': 'Art of Listening', 'latitude': 34.070223, 'longitude': -118.440193 }  
humanities\_classroom = { 'name': 'Art History', 'latitude': 34.071528, 'longitude': -118.441211 }  
  
john\_student = { 'name': 'John Wilson', 'latitude': 34.069149, 'longitude': -118.442639 } # engineering  
jane\_student = { 'name': 'Jane Graham', 'latitude': 34.069601, 'longitude': -118.441862 } # geology  
pam\_student = { 'name': 'Pam Bam', 'latitude': 34.071513, 'longitude': -118.441181 } # humanities  
  
student\_list = [john\_student,jane\_student,pam\_student]  
classroom\_list = [geology\_classroom,psychology\_classroom,music\_classroom,humanities\_classroom,engineering\_classroom]  
  
Call:  
  
studentsInClasses(student\_list, classroom\_list)  
  
Returns:  
  
[{'latitude': 34.069149, 'name': 'John Wilson', 'longitude': -118.442639}, {'latitude': 34.069601, 'name': 'Jane Graham', 'longitude': -118.441862}, {'latitude': 34.071513, 'name': 'Pam Bam', 'longitude': -118.441181}]  
  
  
  
  
  
  
  
Example 2:  
  
Input Variables:  
  
engineering\_classroom = { 'name': 'Principles of computational geo-location analysis', 'latitude': 34.069140, 'longitude': -118.442689 }  
geology\_classroom = { 'name': 'Sedimentary Petrology', 'latitude': 34.069585, 'longitude': -118.441878 }  
psychology\_classroom = { 'name': 'Introductory Psychobiology', 'latitude': 34.069742, 'longitude': -118.441312 }  
music\_classroom = { 'name': 'Art of Listening', 'latitude': 34.070223, 'longitude': -118.440193 }  
humanities\_classroom = { 'name': 'Art Hitory', 'latitude': 34.071528, 'longitude': -118.441211 }  
  
john\_student = { 'name': 'John Wilson', 'latitude': 34.069849, 'longitude': -118.443539 } # engineering  
jane\_student = { 'name': 'Jane Graham', 'latitude': 34.069901, 'longitude': -118.441562 } # geology  
pam\_student = { 'name': 'Pam Bam', 'latitude': 34.071523, 'longitude': -118.441171 } # humanities  
  
classroom\_list =   
[geology\_classroom,psychology\_classroom,music\_classroom,humanities\_classroom,engineering\_classroom]  
  
student\_list2 = [john\_student,jane\_student,pam\_student]  
  
Call:  
  
studentsInClasses(student\_list2, classroom\_list)  
  
Returns:  
  
[{'latitude': 34.071523, 'name': 'Pam Bam', 'longitude': -118.441171}]  
  
  
  
  
  
  
  
  
  
  
**Bonus**  
  
Perform the following in Python.

Story: We need to implement a way of managing student crowd control so we want an additional function that returns only students found in classrooms where there are clusters of 2 or more students, the signature would be the same but called studentClustersInClasses  
  
  
**Additional Expectations**  
  
Write two unit tests for the above two cases as well as write an additional unit test that you think would be helpful to verify.  
  
This problem should take no more than 2-4 hours, please place your code on GitHub and share with us. If you have any questions feel free to contact us.