**Amazon Restaurants and Tickets Onsite Interview Practice Problems**

1)

Implement a function to return the N highest rated movies that are considered similar to a given movie. Each movie has a rating and a list of movies it is most similar to. The full list of recommendations for a given movie is its entire similarity network (similarities, similarities of similarities, etc.).

The order of the returned movies does not matter. The original movie **should not** be in the list of recommendations. You may assume the number of recommendations to return will never be negative, but may be zero.

Additionally, please provide and explain (in comments) the runtime complexity of your solution.

Example:

A(Rating 1.2)

/ \

B(3.6) C(2.4)

\ /

D(4.8)

In the above example, edges represent similarity and numbers indicate a movie's rating.

* getMovieRecommendations(A, 2) should return B, D (sorting order doesn't matter so it can also return D, B).
* getMovieRecommendations(A, 4) should return B, C, D (it can also return these in any order eg: D, C, B). Note that the original movie (A) is not included despite numTopRatedMovies being large enough.
* getMovieRecommendations(A, 1) should return D. Note that the distance from A to D doesn't matter; the high est rated movies in the network should be returned regardless of distance from the original.

Finally, you should design a solution which operates as a library, and does not make assumptions about the code calling it. For example, you may not assume that the movie network will remain the same between calls to the getMovieRecommendations method.

2)

The goal of this exercise is to design a highly scalable and available "Street Address" webservice.  This service exposes REST-based APIs to create, read, update, and delete addresses used by customers (i.e., an address book). While conceptually simple, this service plays a key role in the core of an ecommerce platform. For example, we need to know where to ship a customer order. A street address is also needed to validate a new payment instrument, like a credit card. Therefore, this service is critical and will be called at a high rate by a large number of clients.

For the purposes of this exercise, focus mainly on scalability and availability. Don’t worry as much about performance, as long as it’s reasonably accounted for in the design. Also, assume that the requests to this service are 50% reads and 50% writes. Thus designing a system optimized for reads does not meet the goal of this exercise.

The following is an example of an API, specifically Read.

1) List<String> getAddress(Customer ID) - > return entire address book of the specified customer

2) String getAddress(Customer ID, Address ID) -> return a specific address for the customer (e.g., home, work, Uncle Bob, etc.)

The following is the simplest architecture for this webservice – one host running the business logic and another host running the persistence (e.g., relational database). However this architecture will not scale and is not highly available (i.e., many bottlenecks and single points of failure). How can this base architecture be updated (staying at the block diagram level) so that the resulting system will scalable and be highly available?

------------------------

|  Business Logic |

------------------------

               |

------------------------

|      Database      |

------------------------