MongoDB Laboratory

This laboratory is dedicated to MongoDB, a document data store that belongs to the NoSQL family.
The main goal for this laboratory is to gain familiarity with a typical MongoDB use case. The interested student can understand the details of MongoDB by "playing" with the command line interface that interacts with the MongDB server. On the official website (http://www.mongodb.org/) it is possible to find the list of commands supported by MongDB, and additional useful documentation.

Installing and running the MongoDB server

MongoDB can be downloaded from the following webpage: https://www.mongodb.org/downloads
Put the downloaded file wherever you prefer, open a terminal, go to the directory where you put the downloaded file, then type the following commands

```
tar -xzf mongodb-linux-x86_64-2.6.5.tgz
cd mongodb-linux-x86_64-2.6.5
sudo mkdir –p /data/db
sudo ./bin/mongod
```

The last command will launch the server, and MongoDB is ready to accept connection on port 27017.

Interacting with the MongoDB server

In order to interact with the MongoDB server, a client is necessary. The client will connect to the server on port 27017 and it will send the commands through the connection. In this lab, we will use the Java client; these are the steps to follow for using such a client:

- Download the jar of the Jedis client: http://central.maven.org/maven2/org/mongodb/mongo-java-driver/2.12.4/mongo-java-driver-2.12.4.jar
    - alternatively, check this page http://docs.mongodb.org/ecosystem/drivers/java/
- Put the jar in the same directory of your Java program;
- In your Java program, import the Jedis class:
  ```
  import com.mongodb.*;
  ```
- In your Java program, use the Jedis client, e.g.:
  ```
  MongoClient mongo = new MongoClient("localhost", 27017);
  ```
- Compile your Java program:
  ```
  javac -cp mongo-java-driver-2.12.4.jar myProgram.java
  ```
- Run your Java program (that includes the Redis client):
  ```
  java -cp '.: mongo-java-driver-2.12.4.jar' myProgram
  ```

The student can use as a starting point the file "ecommerce_mongodb.java" provided on the course web page.
For additional information, check this web page:
http://docs.mongodb.org/ecosystem/tutorial/getting-started-with-java-driver/

1 Exercise – Product Catalog

We will see the basic patterns and principles for designing an E-Commerce product catalog system using MongoDB as a storage engine. Product catalogs must have the capacity to store many differed types of objects with different sets of attributes. These kinds of data collections are quite compatible with MongoDB's data model: we will use a single MongoDB collection to store all the product data. MongoDB's dynamic schema means that each document need not conform to the same schema. As a result, the document for each product only needs to contain attributes relevant to that product.

To this aim, at the beginning of each document, the schema must contain general product information, to facilitate searches of the entire catalog. Then, a details sub-document that contains fields that vary between product types. Consider the following example document for an album product.

```
{
  type: "Audio Album",
  title: "A Love Supreme",
  description: "by John Coltrane",

  shipping: {
    weight: 6,
    dimensions: {
      width: 10,
      height: 10,
      depth: 1
    },
  },

  pricing: {
    list: 1200,
    retail: 1100,
    savings: 100,
    pct_savings: 8
  },

  details: {
    title: "A Love Supreme [Original Recording Reissued]",
    artist: "John Coltrane",
    genre: [ "Jazz", "General" ],
    tracks: [
      "A Love Supreme Part I: Acknowledgement",
      "A Love Supreme Part II - Resolution",
      "A Love Supreme, Part III: Pursuance",
      "A Love Supreme, Part IV-Psalm"
    ],
  },
}
```

A movie item would have the same fields for general product information, shipping, and pricing, but have different details sub-document. Consider the following:

```
{
  type: "Film",
  ...,

  shipping: { ... },

  pricing: { ... },

  details: {
    title: "The Matrix",
    director: [ "Andy Wachowski", "Larry Wachowski" ],
    writer: [ "Andy Wachowski", "Larry Wachowski" ],
    ...,
    aspect_ratio: "1.66:1"
  },
}
```

For most deployments the primary use of the product catalog is to perform search operations. These exercises provide an overview of various types of queries that may be useful for supporting an e-commerce site.

The exercises are inspired by the following example: http://docs.mongodb.org/ecosystem/use-cases/product-catalog/

1.1 Exercise 1

Problem statement: Find Albums by Genre and Sort by Year Produced.

Hint: To support this query, create a compound index on all the properties used in the filter and in the sort.

1.2 Exercise 2

Problem statement: Find Products Sorted by Percentage Discount Descending. While most searches will be for a particular type of product (e.g album, movie, etc.,) in some situations you may want to return all products in a certain price range, or discount percentage.

Hint: To support this type of query, you will want to create an index on the pricing.pct_savings field

1.3 Exercise 3

Problem statement: Find Movies Based on Starring Actor.

Hint: To support this query, you may want to create indexes…