Exercise 1

1. How does the Flash Translation Level (FTL) work in SSDs?
    a. The FTL maps the logical addresses in the host with the physical addresses inn the flash (wear levering). This is because writing to the same flash can cause errors in the future. Therefore the FTL is making sure that all of the flashes are used as much as possiable to get the maximum effect and "lifetime" of using flash storage.
2. Why are sequential writes important for performance on SSDs?
    a. This is because its easier for the controller to read data that is clustered together. The controller receives an address and
3. Discuss the effect of alignment of blocks to SSD pages.
    a. Write requests should be aligned to clustered pages so that writing to disk can be done with no further write overhead. In the case that the write order excedes the clustered page it will have to read the rest of the content in the last clustered page and combine
    it with the updated data before all the data can be written to a new page.
4. Describe the layout of MemTable and SSTable of RocksDB
    a. the MemTable is an in-memory cache where when a record is written to the database it is immediately added to the Memtable sorted by the key. When the cache capacity is reached it's flushed to the disk as a SSTable. The SSTable are levels of storage with different levels of capacity. When the cache is flushed and stored at the lowest level this level can be filled up quickly. When this happends a compaction is triggered and this SSTable is moved to the next level with "more" storage.
5. What happens during compaction in RocksDB?
    a. The compactor takes a SSTable that has reached maximum capasity and merges it with the SSTable at a higher level with more storage and preferable overlapping key ranges. After the merge we have to sort the keys by order and delete old records to save space. Finally we have to create a new SSTable to replace the filled one.

6. Give some reasons for why LSM-trees are regarded as more efficient than B+-trees for large volumes of inserts.
    a. When writing to a B+-tree with a large volume there is a lot of new leafes and splitting of leafes that needs to happen to make sure that all the keys are in sorted order. This can cause a lot of overhead and delay when inserting larger volums as opposed to LSM-trees where instead the nodes are just merges with higher order nodes. This is in theory faster since merging data is much simpler then trying to place each element in a tree and having to splitt and extend the tree to make room for large data.
7. Regarding fault tolerance, give a description of what hardware, software and human errors may be?
    a. Hardware errors is caused then the servers are damaged, looses power or the ethernet cable is loose. This is usually due to a physical error in the machine or with external cables. Over time these errors are reduced my having multiple servers running where one server can take the extra load if another faults.
    b. Software error are usually bugs in the code, garbage collection is not working properly, third party systems don't work as anticipated or at all. These fault are caused by the system itself and can go unnoticed until the correct circumstance occur and crashes the entire system.
    c. Lastly we have human error where the error lies in the design and build of both the hardware and the software systems. These errors can be configuration, have forgotten to plug in the cable, forgotten to optimize for heavy processes etc.

8. Compare SQL and the document model. Give advantages and disadvantages of each approach. Give an example which shows the problem with many-to-many relationships in the document
model.
    a. An SQL database or a relation database is used when we want to store data in a structured way. This makes it easier for creating relations between data and querying this data by joining multiple tables to obtain the information needed. But because of this strictness its not efficient when trying to scale or dynamically change how the data is presented.

b. Document models on the other hand has not that many rules it has to follow it is extremely fast at writing large amount of changing data. It is also very good at scaling with the datasets. However because of the lack of restrictions its also a lot harder to query this data by f.ex. joining multiple document or obtain the many-to-many relation a lot of applications need.

c. Document based models have a hard time with many-to-many relations where a document contains data in a f.ex. JSON format. Where there is a key and a value. This value may be another document so that a one-to-many relation is quite effictive

d.

The main difference between SQL and document model is that the SQL database will have a knows structure and therefor its possible to have different tables connected to each other with a key. However a document model is much less strict and the developers have to implement the rules and "strictness" off the database instead. There is also the matter of finding data in document vs SQL. Where SQL databases are usually indexed sorted by table its a lot faster to find elements and connect these elements with a JOIN operation. In document models the is no way of JOINING these documents and a many to many relation will be a lot more difficult to obtain. It is possiable but instead you will get an infinite loop where they keep refering to each other and the object will become impossiably large.

```
paper: {
  name: paper1,
  section: 22,
  words: 12345,
  authors: [
    {
      name: Zaim,
      address: Slottet,
      papers: [
        {
          name: paper1,
          authors: [...]
        }
      ]
    }
  ]
}
```

9. When should you use a graph model instead of a document model? Explain why. Give an example of a typical problem that would benefit from using a graph model.
    a. The graph model solves the problem that documents models have by managing many to many connections. Since graphs can in theory have a connection to all the other nodes this is quite powerful when we need to structure a sosial network or a roadmap. A node is the road intersection and a vertex is the road itself.
10. Column compression: You have the following values for a column:

Aa Name	a
values	43 43 43 87 87 63 63 32 33 33 33 33 89 89 89 33
Bitmap	
values = 32	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
values = 33	0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1
values = 43	1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0
values = 63	0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0
values = 87	0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
values = 89	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0
Run-length	
values = 32	7, 1
values = 33	8, 4, 3, 1
values = 43	0, 3
values = 63	5, 2
values = 87	3, 2
values = 89	12, 3

10. Communication
    a. In the case of MessagePack there really should not be any problem adding another attribute where we are sending both the type and name of the attribute in the byte sequense. Therefor adding another attribute should be both forward and backward compatible.

b. With Apache Thrift and Protocol Buffers we need to add a number and a optional attribute to the schema or add a default value to the attribute. This should be both forward and backward compatible since both techniques just skip unknows attribute there should not be a problem editing the schemas.

c. Avro there should be no problem adding the attribute but this needs to have a default value. I believe this should also be both backward and forward compatible as long as the writer and reader schamas are compatible. If there are any unknown attribues these will just be set to null.