Question 1
Consider the following database class:

```
import random
RANGE=10**6  # you can change this param for demo purposes
ELEM_NO = 10**4 # ditto


class DB(object):
    def __init__(self):
        self.db = random.sample(range(RANGE), ELEM_NO)

    def query(self, elem):
        if elem in self.db:
            return True
        return False
```

Using Bloom filters (can be an external implementation) improve the availability of this database. Compare and report on efficiency of these two constructions (i.e., the original DB and with your improvements) under

- standard load (when queries are for existing objects),
- adversarial load (when queries are for non-existing objects).

Question 2
Design and implement a simple replicated service, where a client can contact to a replica server and can execute "insert" and "get" commands:

- insert (**key**, **value**),
    - when (**key**, **value**) is inserted it has to be replicated with all (alive) replica servers
    - if **key** exists already, the **value** should overwrite the old value (this change has to be replicated too)
- get **value** for a given **key**, or **None** if there is no value for the **key**

Keys and values are strings or bytes. Communication between clients and replicas is TCP or UDP. For replication use a package based on a consensus protocol (e.g., `PySyncObj`). Run a few replica servers and demonstrate that crashing some of them does not stop the service.