Homework 3

CS425/ECE428 Spring 2019

Due: Monday, April 8 at 12:00 **NOON**
NO LATE SUBMISSIONS ACCEPTED

1. Consider a Chord system with 12-bit identifiers. It has the following nodes, listed in hex and decimal for your convenience:

```
014, 017, 066, 06b,       20,   23,  102,  107
095, 0a6, 0a9, 0c0,      149,  166,  169,  192
0dd, 105, 147, 153,      221,  261,  327,  339
15e, 175, 17f, 1dc,      350,  373,  383,  476
1f2, 21e, 27f, 2de,      498,  542,  639,  734
353, 3a4, 3bf, 3ce,      851,  932,  959,  974
403, 416, 442, 456,     1027, 1046, 1090, 1110
45c, 464, 483, 4aa,     1116, 1124, 1155, 1194
4ca, 4e8, 522, 539,     1226, 1256, 1314, 1337
55e, 571, 60d, 658,     1374, 1393, 1549, 1624
67b, 689, 6d3, 6f1,     1659, 1673, 1747, 1777
6fb, 712, 738, 741,     1787, 1810, 1848, 1857
749, 74f, 7d1, 7e9,     1865, 1871, 2001, 2025
849, 865, 8f2, 91b,     2121, 2149, 2290, 2331
926, 984, 996, 9b8,     2342, 2436, 2454, 2488
9b9, 9c1, a21, a43,     2489, 2497, 2593, 2627
a5c, a62, a6a, a72,     2652, 2658, 2666, 2674
a92, aac, ac1, ada,     2706, 2732, 2753, 2778
b01, b17, b21, b78,     2817, 2839, 2849, 2936
bb0, bcc, bd9, bdc,     2992, 3020, 3033, 3036
bf7, c18, c1b, c98,     3063, 3096, 3099, 3224
caf, cc0, ce5, d13,     3247, 3264, 3301, 3347
d9c, e0c, e27, e49,     3484, 3596, 3623, 3657
e63, e6d, e7a, edc,     3683, 3693, 3706, 3804
f4a, f5d, f71, fd6,     3914, 3933, 3953, 4054
```

   (a) (4 points) List the finger table of node 0x926 (2342)

   (b) (4 points) List the nodes that 0x926 (2342) would contact during a lookup of the key 0x123 (291)

   (c) (4 points) Identify the nodes that will store the largest expected number of keys and the smallest. (Assume for now that a key is stored at only the successor node.) What is the ratio of their expected storage?

   (d) (4 points) A power outage takes out half the nodes: the ones with even identifiers. Assume no stabilization algorithm has had a chance to run, and so the finger tables have not been updated. List the nodes that 0x7e9 (2025) would contact to look up the key 0x480 (1152). (When a node in the normal lookup protocol tries to contact a finger entry that is no longer alive, it switches to the next best finger that is alive.)

2. (a) (8 points) Use an RPC compiler, such as Apache Thrift, to answer this question. Write down an interface specification for a reader/writer locking service. Your API should allow you to create a new lock and then lock/unlock it for reading and writing.

   Use the RPC compiler to generate an implementation of your protocol. Include in your submission the code for your interface definition (with comments), and a page each of the generated stub and skeleton files.

   (b) (2 points) Identify a function in the C, Go, Python, or Java standard library that has a side effect but is idempotent. Briefly explain your answer.

3. (a) (3 points) Consider a Raft cluster with five nodes, with logs as described by follows. Each event in a log is denoted by a letter; different letters represent different events, and the subscript indicates the term of the event.

   - S_1: committed: a_1, b_2, c_2, uncommitted: d_3
   - S_2: committed: a_1, b_2, uncommitted: c_2, d_3, e_6
   - S_3: committed: a_1, b_2, uncommitted: c_2
   - S_4: committed: a_1, b_2, uncommitted: c_2, d_3, e_6, f_6
   - S_5: committed: a_1, b_2, c_2, uncommitted: g_5, h_5, i_5, j_5, k_5

   Which of the five nodes could be elected leader? Explain.

   (b) (2 points) Is d_3 guaranteed to be eventually committed? Explain; you may need to offer a sequence of events.

   (c) (3 points) Not related to the previous question, describe a sequence of events where three nodes (out of five) could be in the leader state.

4. (a) (4 points) Consider the following transaction (T1):

```
x = a.getbalance()
y = b.getbalance()
c.withdraw(x-y)
a.deposit(x-y)
```

   List when the locks on each of the objects are acquired or upgraded, and what type of lock is acquired. (x and y are local variables to the transaction)

   (b) (3 points) Consider a second transaction (T2):

```
z = b.getbalance()
w = c.getbalance()
c.withdraw(z-w)
b.deposit(z-w)
```

   (c) (3 points) Show an interleaving of T1 and T2 that is serially equivalent, but impossible under two-phase locking (strict or reader/writer)

   (d) (3 points) Show an interleaving of T1 and T2 that is impossible with strict two-phase locking but possible with non-strict locking (reader/writer)

   (e) (3 points) Suppose that instead of lock upgrades, transactions released a read lock and then acquired a write lock. Show a non-serially equivalent interleaving that would be possible in this situation.