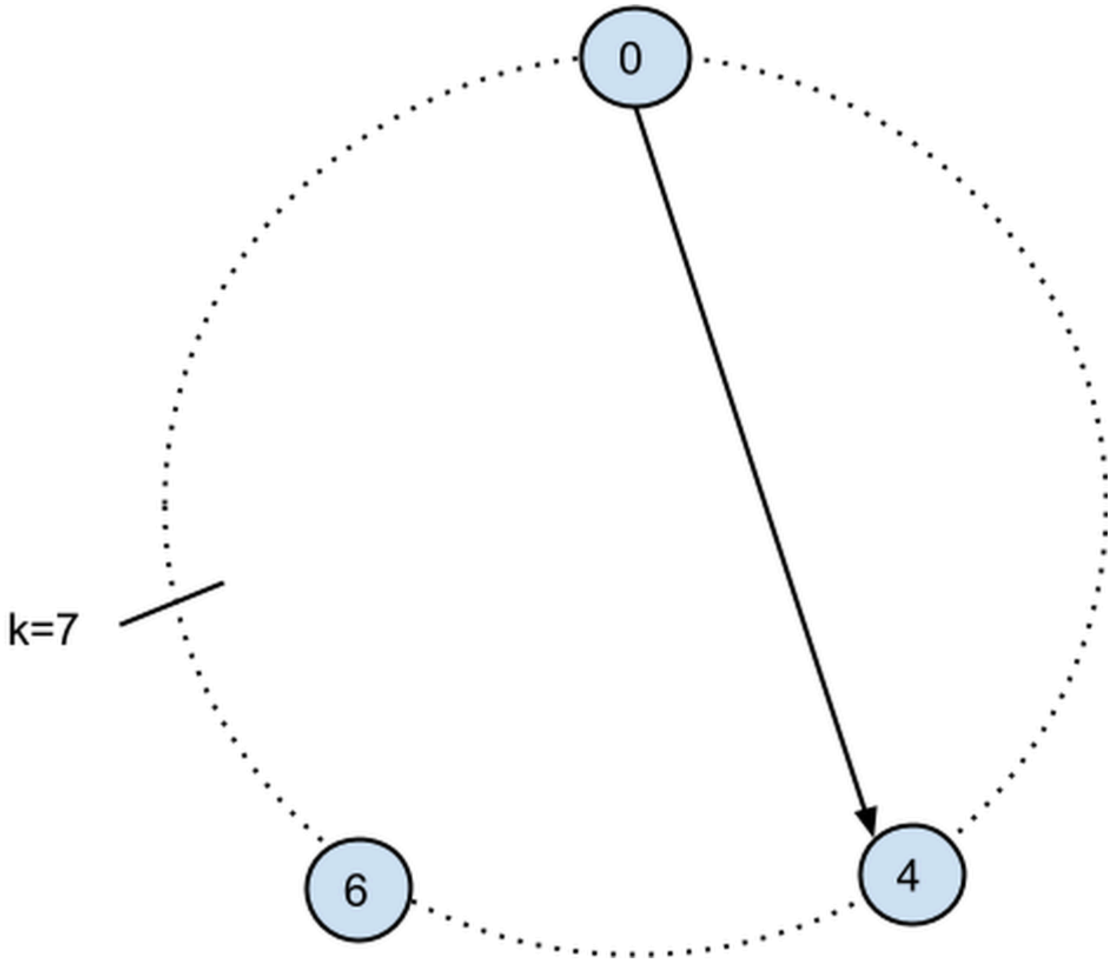
Justin Williams ID# 20365214
Vincent Coste ID# 20365463

1.



If m = 3 then there are n = 8 nodes. Say p = 0, say k = 7 then q = FTp[m] = 4. If there are no peers between p and q, p - q = 0. Further say that there is a peer at 6 then k - p = 1.
The equation: q - p (k - p) / 2 is proved wrong.

2.
a) Every finger table has m entries, if every node is a peer then for a peer there will be m other peers with a finger table entry pointing to that. Therefore if one peer is removed m tables will have to be updated. This also means that there are m - 1 finger tables, the more nodes that are removed, the less finger tables there will be. This works for the reverse, the worst case number of tables that have to be updated for adding a new node is m.
b) Since each node has one entry, that points to p+1, every node has one other node pointing to it. This means that adding a new node will have a worst case of requiring one finger table to get updated.

3.
For an m bit identifier, if every node is a peer, ie n = 2m, then the worst case scenario for the number of jumps from a peer p to peer k would be log(n) = log(2m) = m, which turns out being a linear function of m.

4. Let xi and xj be the coordinates of a peer in the nxn matrix. We need to map them to the range: [0, n2-1]. An invertible function to do this could be:
f(n) = (xi-1) + (xj-1)n
This function will map coordinates xi=1, xj=1 to zero: (1-1) + (1-1)n = 0 + 0 = 0 and map coordinates xi=n, xj=n to zero: (n-1) + (n-1)n = n – 1 + n2 – n = n2 – 1 which is what we wanted.

5. We know that P and Q are neighbors of R and that each node can choose randomly c neighbors. N is the total number of nodes in the graph. The probability that P and Q are neighbors is:
P (P and Q are neighbors) = (number of possibility to pick Q from P)/(number of possibility to pick (c-1) neighbors from P)
P (P and Q are neighbors) = $\frac{\binom{N-3}{N-2}}{\binom{N-2}{c-1}}$

6.
Bully:
We assume that the node with process ID (n+1) crashes. In the worst case it will be process ID 1 that notices that it crashed. Once this occurs, it should start an election by send Election messages to every process with higher process ID (ie all other processes), this represents n requests. Then it will receive (n-1) OK messages because there is one process down (with process ID (n+1)). This will continue until the node with process ID n is elected. The total number of messages sent is as follows:

$$\sum_{i=0}^{n-1}[(n-i)+(n-i-1)]$$

Once process ID n is selected, it has to send n+1 Coordinator messages
We also know that the communication cost is constant, let's name it c.
Then the worst-case communication cost is
(n+1) + c$\sum_{i=0}^{n-1}[(n-i)+(n-i-1)]$

Ring:
In the worst-case process 1 notices that (n+1) is down. It then sends an election message to the next peer with its ID, in turn it will add to the list of IDs it's own ID and send it to the next peer. Then the worst-case communication cost is:

$$\mathrm{n}\times\mathrm{c}\times\sum_{i=1}^{n} i + \mathrm{c}\times\mathrm{n}$$

7. To cancel wasteful election messages, the peer that starts the election should keep track that it started the election with a Boolean flag and of the time at which it started it. The message sent should have a timestamp for when the first element of the list started the election process. When the message reaches another peer that started an election, the message is passed along if the time at which the election was initiated is prior to the one of the current peer.