Muthu Chidambaram
CS4501
Problem Set 2

Problem 1.

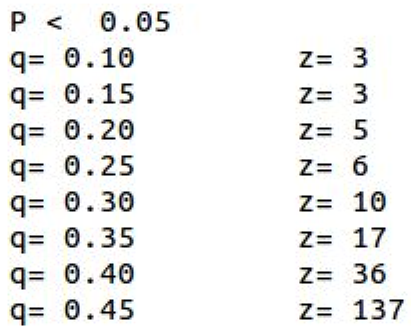
a. A greedy attacker with a majority of the mining power will only be incentivized to play by the rules if his reward for mining honestly is greater than simply double-spending his previous transaction outputs. Therefore, Satoshi assumes that the mining reward (or transaction fees, if the mining reward has gone to 0) will be large enough to warrant honest mining at any given point in the life of the blockchain.

Problem 2.

a. Using Satoshi's definitions, it means that given that the probability the attacker finds the next block is q=0.45 and that he is z=340 blocks behind, the probability that he catches up is less than 0.001.

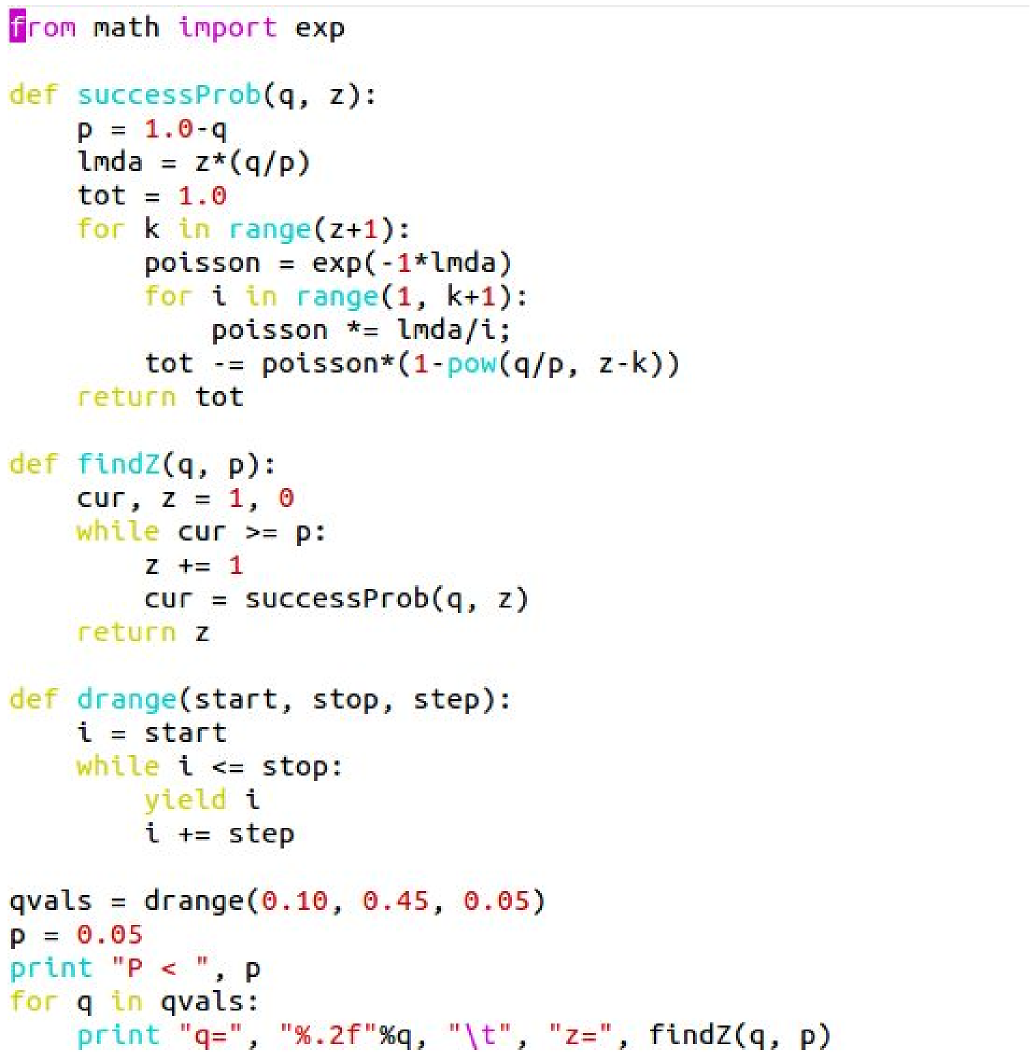
Problem 3.

a. The z-values for the associated values of p and q would be as follows:



```
P <  0.05
q= 0.10          z= 3
q= 0.15          z= 3
q= 0.20          z= 5
q= 0.25          z= 6
q= 0.30          z= 10
q= 0.35          z= 17
q= 0.40          z= 36
q= 0.45          z= 137
```

b. The code I wrote to generate these values is shown below:



```
from math import exp

def successProb(q, z):
    p = 1.0-q
    lmda = z*(q/p)
    tot = 1.0
    for k in range(z+1):
        poisson = exp(-1*lmda)
        for i in range(1, k+1):
            poisson *= lmda/i;
        tot -= poisson*(1-pow(q/p, z-k))
    return tot

def findZ(q, p):
    cur, z = 1, 0
    while cur >= p:
        z += 1
        cur = successProb(q, z)
    return z

def drange(start, stop, step):
    i = start
    while i <= stop:
        yield i
        i += step

qvals = drange(0.10, 0.45, 0.05)
p = 0.05
print "P < ", p
for q in qvals:
    print "q=", "%.2f"%q, "\t", "z=", findZ(q, p)
```

Problem 4.

a. With this scheme, it is not easy to determine whether or not the data obtained from data[i] was actually the data that was in the ith record.

b. No, we cannot be sure that this is indeed the data item we wrote because again, data could have been moved around within the database itself.

Problem 5.

a. The write procedure involves concatenating the new string and recomputing the hash of the new fully concatenated string. The read/verify procedure involves concatenating all of the strings in the database and computing the hash of the resulting string, and then checking to make sure this hash matches the locally stored hash.

b. **Writing is a linear time operation**, since if data is removed or added then the full concatenated string has to be recomputed and then hashed (assuming hashing is a constant time operation). **Reading is also linear time**, since the data in the database also has to be concatenated prior to being hashed. **Verification is constant time**, since it only involves comparing the database hash to the local hash.

Problem 6.

a. Using a merkle tree, write operations simply become adding the hash of the new data to the appropriate branch of the tree and then updating the hashes of all parent nodes up to and including the root node. Reading and verifying data from the database can be done by computing the hash of the ith element and then using the hashed database element along with neighboring elements in the locally stored tree to compute a root hash which can be compared to the stored root hash.

b. **Writing is now a logarithmic operation**, since only the hash data along the path to the root from the new leaf node has to be modified. **Reading and verifying an individual record is now a logarithmic operation** due to having to only consider the path from the element up to the root within the merkle tree. **Verifying all elements is, however, now an N*Log(N) operation** due to each element taking logarithmic time to verify.

Problem 7.

a. Since a block is found every 10 minutes, we expect 1 block/10 min * 60 min/1 hour * 24 hours/1 day = 144 blocks to be found per day. If the mining pool is honest (and everyone else is honest), then its expected number of found blocks for the day is simply alpha*144 = 0.15*144 = **21.6**. If the mining pool is selfish (and everyone else is honest), then its expected number of found blocks can be calculated using the formula derived during class: alpha^2*(2+alpha/(1-2*alpha))*144 = **7.174**.

b. Since there are 1440 minutes in a day, we expect an honest mining pool to find 21.6/1440 blocks/min, or **0.015*t blocks in t minutes**. Similarly, we expect a selfish mining pool to find 7.174/1440 blocks/min, or **0.00498*t blocks in t minutes**.

Problem 8.

a. A block is orphaned if the pool finds the block, but the rest of the network had already found the block during some time window less than or equal to L prior to the pool. Thus the number of blocks orphaned by the pool is then the probability the pool finds the block and the network finds the block, multiplied by the number of blocks mined in L seconds. Since 0.1 blocks are found per minute (assuming that a block is found every 10 minutes), 0.00166 blocks are found every second. Therefore, we expect the number of orphaned blocks to be **alpha*(1-alpha)*L*0.00166**.

Problem 9.

a. If the pool mines selfishly and the rest of the network mines honestly, we can no longer use our simple alpha and (1-alpha) model. The pool is now expected to find alpha^2*(2+alpha/(1-2*alpha)) blocks per cycle, and the rest of the network is expected to find 1-alpha^2 blocks per cycle. We can still use this model, since we know that the selfish pool has a low-latency connection that allows it to announce withheld blocks immediately. Thus, we expect the number of orphaned blocks to be **alpha^2*(1+alpha/(1-2*alpha))*(1-alpha^2)*L*0.00166**.

Problem 10B.

a. What is the current state of national and global Internet regulation? Should I be worried about censorship or potential limitations in the near future?

b. How has the rise of bitcoin and similar cryptocurrencies affected/influenced Internet-related legislation?

c. What is the procedure for tracing and capturing the perpetrators of a bitcoin ransom?

Problem Challenge 2.

a. We now have to factor in propagation time in our model. This means that we can no longer simply use 2 blocks as the release state - the release state must be determined based on the time it takes for the published blocks to propagate across the network. In other words, we release all withheld blocks when our lead is cut to **N** blocks, where N is the smallest number of blocks that can propagate across the network in a time less than what it would take the rest of the network to mine that many blocks.