Question 1

Hash functions and proofs of work

Let $H : P \times S \to \{0, 1, ...2^n - 1\}$ be a collision resistant hash function.

Let H' be a new hash function, $H' : P \times S \to \{0, 1, ...2^{n'} - 1\}$ (where $n' = n + \log(d)$), defined as:

$$H'(p, s) = (num(s) \bmod d) \mid H(p, s)$$

Here $num(s)$, denotes the number representation of s. In other words the new hash function is defined by append $H(p, s)$, to modulo of s wrt d.

Claim 1**:** H' is collision resistant hash function.

Proof. Suppose H' is not a collision resistant hash function, ie. its easy to find x, y, such that $H'(x) = H'(y)$ and $x \neq y$. Which means that last n bits of $H'(x)$ are same as that of $H'(y)$, or $H(x) = H(y)$. This contradicts or assumption that H was collision resistant. ■

claim 2**:** H' is not proof-of-work secure.

Proof. Note that for every puzzle $p \in P$, $H'(p, 0) = (0 \bmod d) \mid H(p, 0) < \dfrac{2^{n'}}{d}$.

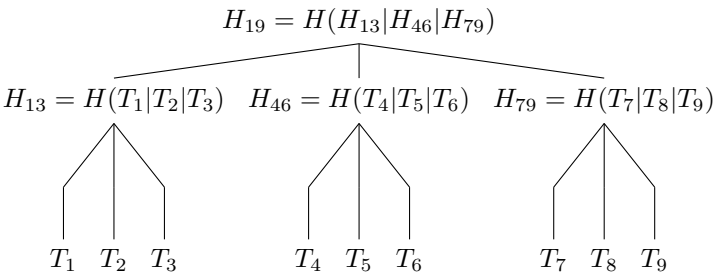
For a fixed difficulty(d), it's trivial to find a solution, or H' is not proof-of-work secure. ■

Therefore, there is a collision resistant hash function that is not proof-of-work secure.

Question 2

Beyond binary Merkle trees

(a)



Commitment tree for $S = \{T_1, T_2, ...T_9\}$
Each node of the tree stores the hash of it's 3 children concatenated. Finally Alice keeps the hash value of root of merkle tree.
To prove membership of T_4, Alice need to provide values of T_5, T_6, H_{13}, H_{79}. Bob verifies them by computing H_{46} using T_4, T_5, T_6, and then computes H_{19} using computes H_{46} and provided H_{13}, H_{79}, and compare with the root hash stored earlier.

(b)

Note that height of above merkle tree as a function of n, k is $\lceil \log_k n \rceil$ For each level we need to store $k - 1$ for proof.
Length of proof to prove existance of T_i is $(k - 1)\lceil \log_k n \rceil$, in above example it is $2\lceil \log_3 n \rceil$

(c)

In a binary tree $k = 2$, ie proof size = $\lceil \log n \rceil$, while for a general k-ary tree it is $(k-1)\lceil \log_k n \rceil$

Their ratio is:

$$\frac{(k-1)\lceil \log_k n \rceil}{\lceil \log n \rceil}$$

ignoring the ceil function we get:

$$\frac{(k-1)\log_k n}{\log n} = \frac{(k-1)\log 2}{\log k} >> 1 \text{ as } k \text{ increases}$$

One advantage of this over the regular binary merkle tree can be that we would need to compute very few hash , ie this would be beneficial for constly hash functions. Precisely we need to call H $\lceil \log_k n \rceil$ times, which is approximately $\log k$ times better that binary tree.

Question 3

Hiding vs. binding commitments

Question 4

Bitcoin script

Question 5

Lightweight clients

Question 6

BitcoinLotto