Hash Functions

1. Assume H is collision-resistant. Then an efficient adversary cannot compute any collision $H(x) = H(x')$, $x \neq x'$, a fortiori if x and its hash value $H(x)$ is fixed. This gives second-preimage resistance.
   Now suppose that H is not a one-way function and an adversary can efficiently find preimages. Fix $x \in D$ and set $y = H(x)$. The adversary is given y and computes $x' \in D$ with $H(x') = y$. Since H is not injective, there are many preimages of y and the probability of $x \neq x'$ is significant. Then x' is a second preimage. This shows that a second-preimage resistant hash function is preimage resistant.

2. Suppose a function $f : D \to R$ is given, where $f(x) = l(x) + b$ is affine, $l(x)$ is linear and $|D| > |R|$. Then the kernel of the $GF(2)$-linear map l is nontrivial and a vector $v \in D$, $v \neq 0$ with $l(v) = 0$ can be efficiently computed using Gaussian elimination. We have $f(v) = f(0) = b$, and so f is not collision-resistant.

3. Suppose the hash value would not depend on one of the input bits. Then a collision can be produced by choosing any message and flipping this particular bit. Both messages would have the same hash value.

4. a) No, since the hash value would not depend on the low bit.
   b) and c) Yes, since a collision of the modified hash function would yield a collision of the original hash function. However, we assumed that the original hash function is collision-resistant.
   d) No, since XORing the blocks produces collisions: let H' be the modified hash function k the block length. Choose any nonzero word a of length k with $a \neq 0^k$. Then $H'(a\|a) = H(a \oplus a) = H(0^k) = H'(0^k\|0^k)$. This shows that H' is not collision-resistant.

5. $H(m)$ is defined as the output of the last Merke-Damgård iteration *after padding* the message m. An adversary would append the padding bits $10\ldots0$ and the encoded length L followed by any message m'' and set $m' = 10\ldots0\|L\|m''$. Then they compute $H(m\|m')$ by running Merkle-Damgård iterations on $m''\|10\ldots0\|L'$ with initial state $h(m)$. This is possible with only $h(m)$ without knowing m. However, the adversary needs to know the length L of m and thus the encoded length L' of $m\|m'$.

6. The length padding ensures that a collision-resistant compression function yields a collision-resistant Merkle-Damgård hash function. In other words, a collision in H would give a collision in the compression function: if two inputs have different lengths, then their last blocks are different after length padding. In this case, a collision in H implies a collision in the last Merkle-Damgård iteration. If two different inputs of the same length have identical hash values, then there must be a collision in one of the iterations of the compression function.

7. Choose two keys k, k' with $k \neq k'$ and any $c \in \{0,1\}^l$. Set $m = E_k^{-1}(c)$ and $m' = E_{k'}^{-1}(c)$. Then $c = E_k(m) = E_{k'}(m')$, and we have found a collision $f(k,m) = f(k',m')$.

8. One checks that the values of $Ch(B,C,D)$, defined using XOR and alternatively using OR, are identical for all B, C and D. The same is true for $Maj(B,C,D)$.

9. $\frac{10^9}{512} = 1{,}953{,}125$. With the required padding, 1,953,126 blocks are processed, and this is the number of calls to the SHA-1 compression function needed to compute the SHA-1 hash value. For the 256-bit variant of SHA-2 we get the same number of calls. For SHA3-256 the rate is $r = 1088$ and $\frac{10^9}{1088} \approx 919{,}117.6$. The message is padded and 919,118 Keccak-f calls are necessary to compute the SHA3-256 hash value.
10. If the rate r decreases, the capacity c increases and fewer messages bits are processed in each iteration. The Keccak-f function is applied more often, which may increase the security of the construction. However, it slows down the computation.
11. Let $l \leq 512$ be a fixed SHA-3 output length, $r \geq 576$ the corresponding rate and $r + c = 1600$. We use the SHA-3 padding string $pad = 011001$ and consider input messages $x \in \{0,1\}^{r-6}$ of length $r - 6$. Let $f : \{0,1\}^{1600} \to \{0,1\}^{1600}$ be the Keccak-f function and let $pr : \{0,1\}^{1600} \to \{0,1\}^l$ be the map that extracts the first l bits from a binary string of length 1600 bits. Define
$$g : \{0,1\}^{r-6} \to \{0,1\}^l, \quad g(x) = pr(f(x \| pad \| 0^c)).$$
Then $g(x)$ gives the SHA-3 hash value of $x \in \{0,1\}^{r-6}$. If f were linear then g would be an affine map over $GF(2)$, i.e., $g(x) = Ax + b$, where the matrix A and the vector b can be determined using images of g. Since the domain of g has dimension $r - 6$ and the codomain dimension $l < r - 6$, there are collisions in g. They can be efficiently computed using Gaussian elimination. This would yield a collision in SHA-3.
12. No, since the capacity bits are difficult to control. It is not clear whether a given 1600-bit block occurs as SHA-3 state of any input. However, a collision of the form $f(m_1 \| 0^c) = f(m_2 \| 0^c)$, where m_1 and m_2 are padded messages of length r, would result in a collision of the hash function.