1. (a) Let L be an arbitrary NP language, accepted by a DTM $M(x, y)$.
   Recall the reduction $L \leq_P$ 3-SAT presented in Cook's theorem:
   Given x we construct $f(x) = \phi_{M,x}$ in variables $\{y_i\}_{i=1}^n$ that is satisfiable if and only if $M(x, y)$ accepts. Since the same y is a witness for both $x \in L$ and $f(x) \in$ 3-SAT, we conclude that 3-SAT is NP-hard under Levin reductions as $(f, identity)$ are the requested functions.

   (b) Consider the polynomial self-reduction for SAT presented in class. We needed a SAT oracle, but as A is assumed to be NP-hard, we can polynomially reduce every SAT query to an A query. Thus, there's a polynomial oracle TM M^A that given a formula ϕ, computes a satisfying assignment $\rho(\phi)$ for it, if exists.
   Consider the Levin reduction (f, g) from A to 3-SAT (shown to exist in 1a since $A \in NP$). Given $x \notin L$, we have $f(x) \notin$ 3-SAT so $M^A(f(x))$ rejects; Given $x \in L$, we have $f(x) \in$ 3-SAT so $M^A(f(x))$ computes the witness $w = \rho(f(x))$. Using g, we can compute $y = g(w)$, a witness for $x \in L$.
   Obviously, all the reductions are done in polynomial time.

2. Let L be a PH-complete language. Then, $\exists k \in \mathbb{N} \quad L \in \Sigma_k$ as $L \in PH = \bigvee_{k\in\mathbb{N}} \Sigma_k$. By definition $\Sigma_k \subseteq PH$; since L is PH-hard, every $L' \in PH$ can be reduced to L so $L' \in \Sigma_K$, i.e., $PH \subseteq \Sigma_k$; the result follows.

3. The claim is true.
   We have SAT $\in NP$-complete $\subset PSPACE$-complete, so every $L \in PSPACE$ can be polynomially reduced to SAT and solved in NP, therefore $PSPACE \subseteq NP$.
   We already know that $NP \subseteq PSPACE$; the result follows.

4. (a) Let $v = v_1 - v_2$ and let $V = span\{v\}$. We have $\dim V = 1$ as $v \neq 0$, so

$$\text{Prob}(xv_1 = xv_2) = \text{Prob}(xv = 0) = \text{Prob}(x \in V^\perp) = \frac{|V^\perp|}{|\mathbb{Z}_2^n|} = 2^{\dim V^\perp - n} = 2^{-\dim V} = \frac{1}{2}$$

   Alternative reasoning:
   since $v \neq 0$ there's a coordinate i such that $v^i = 1$. $\{x^i\}$ are independent, so

$$\begin{aligned}\text{Prob}(xv = 0) = \text{Prob}(x^i v^i = \overbrace{\sum_{j\neq i} x^j v^j}^{z}) = \\ = \text{Prob}(x^i = 0)\,\text{Prob}(z = 0) + \text{Prob}(x^i = 1)\,\text{Prob}(z = 1) = \\ = \frac{\text{Prob}(z = 0) + \text{Prob}(z = 1)}{2} = \frac{1}{2}\end{aligned}$$

   (b) Let $x \in \mathbb{Z}_2^n$ be a random vector. Compute $c = Cx$, $b = Bx$, $a = Ab (= ABx)$ and output *true* if and only if $a = c$.
   Time Complexity:
   Thrice we multiply a $n \times n$ matrix by a vector, so it takes $3O(n^2) = O(n^2)$ time.
   Correctness:
   If $AB = C$ then obviously $c = Cx = ABx = a$ for any $x \in \mathbb{Z}_2^n$.

Otherwise, there's some row index i such that $AB_i \neq C_i$. Using 4a with $v_1 = AB_i$, $v_2 = C_i$, we get $a_i \neq c_i$ with probability $\frac{1}{2}$, so

$$\text{Prob}(a = c) = \text{Prob}(a_j = c_j \quad j = 1, \ldots, n) \leq \text{Prob}(a_i = c_i) = \frac{1}{2}$$

5. (a) We use a binary search on the range $[1, m]$.
Initialize $low \leftarrow 1, high \leftarrow m$. A typical round looks like this:
   - Alice has her element list $\{a_i\}_{i=1}^n$ and Bob has his list $\{b_i\}_{i=1}^n$. Both Alice and Bob know $low, high, n, m, k$; they calculate $mid = \lceil \frac{low+high}{2} \rceil$.
   - Alice calculates $a = |\{i | a_i < mid\}|$ and sends it to Bob.
   - Bob calculates $b = |\{i | b_i < mid\}|$ and compares $a + b$ to k.
   - If $a + b > k$, then the answer lies in $[low, mid - 1]$ so Bob sends "low" to Alice and both update $high \leftarrow mid - 1$;
   - If $a + b < k$, then the answer lies in $[mid + 1, high]$ so Bob sends "high" to Alice and both update $low \leftarrow mid + 1$;
   - Otherwise, $a + b = k$, i.e., the answer is mid so Bob sends "stop" to Alice and the protocol ends.

   Communication Complexity:
   On every round $high - low$ is halved so we have $\lfloor \log_2 m \rfloor$ rounds until the protocol ends; Each round costs $O(\log n)$ since Alice sends $0 \leq a \leq n$ and Bob sends one of 3 possible replies. The total communication complexity is $O(\log n \log m)$.

   Correctness:
   Each round maintains the invariant: the answer lies in $[low, high]$ so the algorithm will reach a point where $high = low =$ the correct answer.

   (b)