1. To prove the lower bound in the question, we can view the relevant randomized protocol as a distribution over deterministic protocols, where for each deterministic protocol the public random string is fixed. Since we only allow zero error, each of these deterministic protocols must calculate f exactly.

   Assume that f has a fooling set of size t. It follows that any determenistic protocol computing f has its communication complexity bounded below by $\log t$. Combining this assumption with the idea from the previous paragraph, we can say that whatever deterministic protocol we pick from our distribution, it is guaranteed to have communication complexity $D(f) \geq \log t$. Hence we can conclude that $R_0^{\text{pub}}(f) \geq \log t$.

   An example of a function that has an exponential gap between R_0^{pub} and $R_{1/3}^{\text{pub}}$ is EQ_2. If the outputs of f can be encoded in a matrix of size $n \times n$, we know that the only values for which EQ_2 evaluates to 1 are the values on the diagonal, and they can be used as the fooling set. Since the size of this fooling set is n, we know that $R_0^{\text{pub}}(\text{EQ}_2) \geq \log n$ (by argument above).

   We can significantly decrease the amount of bits exchanged if we're allowed to be wrong on 1/3 fraction of inputs. Our protocol can start by picking a substring of the public random string, denoting it $r \in \{0,1\}^n$. Then, Alice can calculate the dot product of r and her value a (mod 2), send the resulting 1 bit over to Bob, and let Bob take the same inner product and compare the values. If the actual values a and b held by Alice and Bob are equal, this will produce the correct result with probability 1. If the values were not equal, this approach can produce an incorrect result with probablity 1/2. To satisfy the error bound of 1/3, we can repeat this process $k \geq 2$ times for some fixed k, each time using a different substring of the random string, halving the probability of error. Since k is constant, the communication complexity is $O(1)$, and hence we have an exponential decrease from R_0^{pub}.

2. TRIBES function requires that x and y share at least one 1 on every row of the matrix. Solutions:

   (a) The most straightforward way to solve TRIBES(x, y) is for Alice to just send over her entire matrix, which requires n bits, and wait for Bob's answer, giving overall communication complexity $O(n)$. This proves the upper bound for the problem.

   To show the lower bound we can reuse some ideas from the DISJ problem. Recall that the matrix encoding outputs of DISJ has rank 2^k when the set size is k and the universe can be thought of as all bitmasks of such sets of form $\{0,1\}^k$.

   Consider ith rows of $\sqrt{n} \times \sqrt{n}$ matrices x and y, denoting them x_i and y_i. Note that for every row i in TRIBES, we're solving the complement of DISJ(x_i, y_i). Thus we can take the matrix M_{DISJ} ($2^{\sqrt{n}} \times 2^{\sqrt{n}}$) and invert its boolean values component-wise to obtain a matrix that encodes the output of $\vee_j(x_{ij} \wedge y_{ij})$. Since we started of with a DISJ matrix of rank $2^{\sqrt{n}}$, we can show that the rank of the resultant matrix is also $2^{\sqrt{n}}$ by replacing 0's with -1's. Denote the final matrix from the previous step T^*.

   Now we can use T^* to generate a matrix that encodes the output of TRIBES(x, y) on any inputs x, y while also showing that overall rank is 2^n. The key insight here is that we're solving $\sqrt{n}$ co-DISJ problems and taking the product of the results to determine the output of TRIBES.

   We start off with a $2^{\sqrt{n}} \times 2^{\sqrt{n}}$ matrix $T_0 = T*$. As mentioned earlier, this corresponds to TRIBES ran on a single row of size $\sqrt{n}$. We can now transform this matrix to encode the output of TRIBES on two rows of size $\sqrt{n}$. We duplicate every row i in T_0 exactly $2^{\sqrt{n}}$ times. Next, we also duplicate all columns j in T_0 exactly $2^{\sqrt{n}}$ times, obtaining a matrix of size $2^{2\sqrt{n}} \times 2^{2\sqrt{n}}$.

   Next, we produce a matrix T_0^* by duplicating (tiling) the whole matrix T^* vertically and horizontally exactly $2^{\sqrt{n}}$ times. Finally, we take the component-wise product of T_0 and T_0^* and denote it T_1. Note that T_1 encodes the the output of TRIBES on two rows of size $\sqrt{n}$ because it's produced by first considering co-DISJ matrices of each row, and then taking the logical AND of the two matrices. It also still has full rank if we replace 0's with -1's.

Repeating the procedure above $\sqrt{n}$ times in total gives us a matrix $T_{\sqrt{n}}$ that encodes the output of TRIBES on $\sqrt{n} \times \sqrt{n}$ matrices, while also having full rank of 2^n. It follows that deterministic complexity of TRIBES is bounded below by $\Omega(\log 2^n)$ or just $\Omega(n)$.

Combining the lower and upper bound, we get $D(\text{TRIBES}) = \Theta(n)$.

(b) By definition of the problem, to show that $\text{TRIBES}(x, y) = 1$ we only need to find one index j on every row i such that $x_{i,j} = y_{i,j} = 1$. Therefore an all powerful prover just needs to point out these indices on each row. There are $\sqrt{n}$ rows, and each column index takes $\log_2(\sqrt{n})$ bits to represent, giving us an overall bit count of $\sqrt{n} \cdot \frac{1}{2}\log_2 n$. Alice can send over these values to Bob and wait for him to confirm that there is a match. Hence the upper bound on $N^1(\text{TRIBES})$ is $O(\sqrt{n}\log n)$.

We can use the fooling set technique to show the lower bound. Consider a $1 \times \sqrt{n}$ vector e_i^T, which has zeros everywhere except the ith position. Consider a matrix x where each row is a vector e_{i_k} for some $i_k \in \{1, \ldots, \sqrt{n}\}$. Clearly, $\text{TRIBES}(x, x) = 1$ because x has at least one 1 on each row. At the same time, if we permute any of the rows of x by shifting the 1 in that row left or right and define the new matrix x^*, we'll see that $\text{TRIBES}(x, x^*) = 0$ because there is now at least one row where x and x^* do not match up. We can exploit this to generate a fooling set. Let M be the set of matrices that start off as an identity matrix, but have either 1 within some row shifted left or right, or have some rows swapped, or both. Then we can define the fooling set S as:

$$S = \{(M^*, M^*) : \text{distinct } M^* \in M\}$$

Clearly, this a 1-fooling set because for any two distinct pairs $(M_1, M_2), (M_1^*, M_2^*) \in S$, neither (M_1, M_2^*) nor (M_1^*, M_2) can be evaluated to 1 since they must differ in at least one place (we can show by contradiction that if this is not the case, the pairs must be identical). Now for the size of this fooling set: there are $\sqrt{n}$ rows in total, and for each row we have $\sqrt{n}$ different choices for the vector e_i^T. This means that $|S| = \sqrt{n}^{\sqrt{n}}$, and we can obtain a lower bound $N^1(\text{TRIBES}) \geq \Omega(\log \left(\sqrt{n}^{\sqrt{n}}\right))$, or, equivalently, $N^1(\text{TRIBES}) \geq \Omega(\sqrt{n}\log n)$.

Combining the lower and upper bound, we get $N^1(\text{TRIBES}) = \Theta(\sqrt{n}\log n)$.

(c) To show that $\text{TRIBES}(x, y) = 0$, we need to find a single row i where in each position j we have $x_{i,j} \wedge y_{i,j} = 0$. Our all powerful prover can identify this row and show its index to Alice and Bob. Then, Alice can send Bob a bitmask of said row, taking up $\sqrt{n}$ bits in total. Including the 1 bit of Bob's reply, we get the overall complexity of $O(\sqrt{n})$, proving the upper bound for $N^0(\text{TRIBES})$.

The lower bound can be proved by showing a 0-fooling set of size $2^{\sqrt{n}}$. Suppose that our all powerful prover has pointed out some row i, so now we only need to check rows x_i and y_i (each of size $\sqrt{n}$) to make sure they don't share any common 1's. Consider all possible vectors $z \in \{0, 1\}^{\sqrt{n}}$ and define its complement z^c, which has all bits flipped. For each z, define a pair (x, y) where $x = z$ and $y = z^c$. Clearly, $\vee_j(x_j \wedge y_j) = 0$, and by definition of z, for any two distinct pairs $(x, y), (x^*, y^*)$ it must be true that either $\vee_j(x_j^* \wedge y_j) = 1$ or $\vee_j(x_j \wedge y_j^*) = 1$. Since $|\{0, 1\}^{\sqrt{n}}| = 2^{\sqrt{n}}$, we can collect all pairs into a 0-fooling S of size $2^{\sqrt{n}}$ that targets a single row. This means we need to exchange at least $\log 2^{\sqrt{n}}$ bits to check row i, proving the lower bound $N^0(\text{TRIBES}) \geq \Omega(\sqrt{n})$.

Combining the lower and upper bound, we get $N^0(\text{TRIBES}) = \Theta(\sqrt{n})$.

3. CIS_G solutions:

   (a) We can prove the lower bound $\Omega(\log n)$ by finding a fooling set of size n for some G. Consider a complete graph G with n nodes. We can generate a clique-independent-set pair (C, I) by picking a single node x^* and defining $I = \{x^*\}$, then putting the remaining $n - 1$ nodes into the clique C. Clearly, I is an independent set because it only has one node, and C is a clique because any subgraph of a complete graph is also complete.

   Since there are n nodes in total, we can define n distinct clique-independent-set pairs using this procedure. Putting them all into a set S we get a 0-fooling set: for any pair $(C, I) \in S$, $\text{CIS}_G(C, I) = 0$, but for any two distinct pairs $(C, I), (C^*, I^*) \in S$ we have $\text{CIS}_G(C, I^*) = 1$ and $\text{CIS}_G(C^*, I) = 1$ (by definition of pairs in S).

   It follows that the deterministic complexity for CIS_G with our choice of G is bounded below by $\Omega(\log |S|)$, or, equivalently, $\Omega(\log n)$.

   (b) The question wants us to prove that $D(\text{CIS}_G) \leq O(\log^c n)$ implies $D(f) \leq O(\log^c C^D(f))$ for an arbitrary choice of a boolean function f.

   Let M_f be the matrix corresponding to outputs of f. We know that there exists some disjoint cover of M_f using monochromatic rectangles. Let C^* be the smallest such cover. By definition, $C^D(f) = |C^*|$. Now, consider only the 1-rectangles from C^*, denoting that set as C_1^*. Note that $|C_1^*| \leq |C^*|$. We can produce a graph G that encodes our problem using the following procedure. Start off with a disconnected graph G that has a node corresponding to every 1-rectangle in C_1^*. Next, for every row x_i in M_f, if said row intersects some 1-rectangles from C_1^*, connect these rectanges into a clique in G. Repeat until we've considered all rows from M_f. The resulting G only depends on the function f (and not its inputs), so it can be encoded as a part of the protocol. As such, it is known to both Alice and Bob beforehand.

   Now, assume the input for function f is (x, y), where Alice holds x and Bob holds y. Alice looks at row x in M_f, and defines her clique to be all 1-rectangles from C_1^* that intersect row x. Bob looks at row y and defines his independent set to be all 1-rectangles that intersect row y. Note that this set is guaranteed to be indepdendent because all nodes in G were initially disconnected, and we've only connected 1-rectangles that intersected a common row. Since our rectangle cover is disjoint, rectangles that intersect some common row cannot also intersect a common column.

   At this point we have an instance of CIS_G, where G is the graph we generated earlier and the inputs are the clique and the independent set picked by Alice and Bob respectively. By our assumption, there exists a protocol that computes CIS_G in $O(\log^c n)$ bits, where n is the size of the graph. Since our graph G has $|C_1^*| \leq C^D(f)$ nodes, we can safely say that there exists a protocol that computes our instance of CIS_G in $O(\log^c C^D(f))$ bits. We can interpret the output as follows: if we get a 1, then x and y both hit the same 1-rectangle, which is only possible when $f(x, y) = 1$. On the other hand, if the output is 0, then x and y don't share any 1-rectangles, so $f(x, y) = 0$. Since we didn't communicate any extra bits beyond solving the CIS_G problem, we can conclude that $D(f) \leq O(\log^c C^D(f))$.