1. (30 points). An $n$-party functionality is said to be easy in a given model for secure computation, if it can be computed by an $n$-secure protocol using a single round of interaction. (The protocol may be randomized.) Below is a list of functionalities, which are assumed to be deterministic unless mentioned otherwise. Each input $x_i$ is a single bit. For each functionality, determine whether it is easy (1) when the adversary is passive; and (2) when the adversary is active. For each claim of easiness, describe an appropriate protocol. Claims of non-easiness do not need to be justified.

We assume the availability of secure point-to-point channels and broadcast, and restrict the attention to perfect security against computationally unbounded, non-adaptive, rushing adversaries.

(a) $f(x_1, x_2) = (0, x_1)$
(b) $f(x_1, x_2) = x_1 \lor x_2$
(c) $f(x_1, x_2) = (0, x_1 \lor x_2)$
(d) $f(x_1, x_2, x_3) = (0, x_1, x_1)$
(e) $f(x_1, x_2, x_3) = (x_3, x_1, x_2)$
(f) $f(x_1, x_2)$ ignores its input and outputs a uniform pair of bits from the set \{(1, 0), (0, 1)\}.

2. (30 points). In this question we suggest two independent modifications of the usual model for secure computation in the presence of an active adversary. In both modifications, the real model stays the same and only the ideal model is changed in a way that allows the trusted third party (TTP) to directly talk to the simulator.

- Variant A: After receiving inputs $x = (x_1, \ldots, x_n)$ from the parties, the TTP computes $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n; \rho)$. It then sends to each honest party $P_i$ its output $y_i$ and sends to the simulator the vector of all inputs $x$. (Note that, as in the usual ideal model, the simulator can change the inputs of corrupted parties.)
- Variant B: The TTP receives inputs from the parties like a standard TTP, except that it additionally receives a vector of inputs $(y'_1, \ldots, y'_n)$ from the simulator. The TTP then computes $(y_1, \ldots, y_n) \leftarrow f(x_1, \ldots, x_n; \rho)$, sends $y_i$ to each corrupted party $P_i$ and sends $y'_j$ to each honest party $P_j$.

In the following, the “standard notion of security” refers to perfect security against an active adversary, in the same model as Question 1. Please give a “yes” or “no” answer to each of the following 4 questions. In case your answer is “yes”, provide a brief explanation. In case your answer is “no”, specify $f$, $\Pi$ and $t$ that serve as a counterexample, and provide a brief explanation.

(a) Suppose $\Pi$ is a protocol that $t$-securely realizes $f$ under the standard notion of security. Is it necessarily a $t$-secure protocol for $f$ under variant A?
(b) Suppose $\Pi$ is a protocol that $t$-securely realizes $f$ under the standard notion of security. Is it necessarily a $t$-secure protocol for $f$ under variant B?
(c) Suppose $\Pi$ is a protocol that $t$-securely realizes $f$ under variant A. Is it necessarily a $t$-secure protocol for $f$ under the standard notion?
(d) Suppose $\Pi$ is a protocol that $t$-securely realizes $f$ under variant B. Is it necessarily a $t$-secure protocol for $f$ under the standard notion?
(e) Which of your 5 answers to Question 1 (for the case of an active adversary) change under variant A? Which answers change under variant B? For each claimed change, provide a brief explanation.

3. (30 points). Let \( g \) be the 5-party functionality defined by 
\[ g(x_1, x_2, x_3, x_4, x_5) = (0, x_1, x_1, 0, 0), \]
where \( x_1 \in \{0, 1\} \). For each of the following protocols, determine whether it 1-securely realizes \( g \) (against an active adversary). If you claim that it is secure, describe a simulator for each corrupted party. If you claim it is not, describe an adversary that cannot be simulated and briefly explain why it cannot be simulated. All messages are sent over secure channels.

(a) Protocol 1:
   i. \( P_1 \) sends \( x_1 \) to \( P_2 \) and \( P_3 \).
   ii. \( P_2 \) and \( P_3 \) output the message they got from \( P_1 \) (or output 0 in case this message is not a single bit).

(b) Protocol 2:
   i. \( P_1 \) uses the 1-private, 4-party CNF scheme to share \( x_1 \) between the other parties. That is, it first additively shares \( x_1 \) into \( (s_2, s_3, s_4, s_5) \) and then sends to each \( P_i \) all additive shares except \( s_i \).
   ii. Each \( P_i, 2 \leq i \leq 5 \), sends the message it got from \( P_1 \) to both \( P_2 \) and \( P_3 \). Note that each additive share \( s_i \) should appear in 3 of the 4 messages.
   iii. Each of \( P_2 \) and \( P_3 \) computes its output as follows. First, it recovers each \( s_i, 2 \leq i \leq 5 \), by taking the majority of the 3 values received from the 3 parties holding \( s_i \) (possibly including itself). Then it outputs the exclusive-or of the 4 majority values.

4. (10 points). For each of the following claims, determine whether it is true or false. No proofs or explanations are required.

   (a) The 1-private, 6-party Shamir secret-sharing scheme is multiplicative.
   (b) There exists a 2-private, 4-party multiplicative secret-sharing scheme.
   (c) There is a perfectly 1-secure 2-party protocol for the OT functionality.
   (d) There is a perfectly 2-secure 9-party VSS protocol.
   (e) There is a 5-party functionality \( f \) that is known not to admit a protocol which is computationally 3-secure against a passive adversary.