All (non-optional) problems are of equal value.

1. In lecture we saw a fingerprinting scheme to check whether two $n$-bit strings are equal which succeeds with probability $\geq 1 - \epsilon$ and requires only $O(\log n/\epsilon)$ bits of communication. Suppose $x \neq y$ and two parties Alice (who has $x$) and Bob (who has $y$) want to find an index $i$ such that $x_i \neq y_i$. Describe a Monte Carlo adaptive communication scheme that the two parties can use to find such an index, that succeeds with probability $\geq 1 - \epsilon$ and always uses at most $O(\log n \cdot \log n/\epsilon)$ bits of communication.

   Hint: Use binary search. How does the probability of error accumulate as the scheme progresses?

2. In this problem, we will investigate a simpler family of hash functions that satisfies a weaker version of universality (with some extra logarithmic factors), but has other nicer properties useful for certain applications.

   Let $m$ be a given integer. Let $p_1, \ldots, p_k$ be the list of all prime numbers at most $m$. You may assume that this list has been precomputed and you may use the known fact that $k = \Theta\left(\frac{m \log m}{\log U}\right)$ (obtaining really tight bounds for $k$ is the subject of the well-known “Prime Number Theorem”).

   Pick a random index $j \in \{1, \ldots, k\}$ and define $h_j : \{0,1,\ldots,U-1\} \rightarrow \{0,1,\ldots,m-1\}$ by

   $$h_j(x) = x \mod p_j$$

   (a) For any fixed $x, y \in \{0,1,\ldots,U-1\}$ with $x \neq y$, prove that $\Pr_j[h_j(x) = h_j(y)] \leq O\left(\frac{\log m \cdot \log U}{m}\right)$.

   Hint: can you upper-bound the number of distinct prime divisors that a number may have?

   (b) 3SUM is a well-known and important theoretical problem: given three sets of integers $A, B, C$ with $|A| + |B| + |C| = n$, we want to decide whether there exist elements $a \in A, b \in B, c \in C$ such that $c = a + b$. One can solve this problem in slightly faster than $O(n^2)$ time but it is a major open problem whether there is an algorithm that runs in $O(n^{2-\delta})$ time for any fixed $\delta > 0$.

   Prof. X claims to have discovered an $O(n^{1.99})$-time algorithm to solve the special case of the problem when $A, B, C \subseteq \{0,1,\ldots,n^4\}$. Show how to use Prof. X’s algorithm to solve the more general case of the problem when $A, B, C \subseteq \{0,1,\ldots,n^{100}\}$ by a Monte Carlo $O(n^{1.99})$-time algorithm with error probability $\leq 1/4$.

   Hint: Use part (a). The property that $h_j(a) + h_j(b)$ is equal to $h_j(a+b)$ or $h_j(a+b) + p_j$ may be helpful.

3. In lecture we discussed the Karp-Rabin randomized algorithm for pattern matching. The power of randomization is seen by considering the two-dimensional pattern matching problem.
The input consists of a $n \times n$ binary matrix $T$ and a $m \times m$ binary matrix $P$. Our goal is to check if $P$ occurs as a (contiguous) submatrix of $T$. Describe an algorithm that runs in $O(n^2)$ time assuming that arithmetic operation in $O(\log n)$-bit integers can be performed in constant time. This can be done via a modification of the Karp-Rabin algorithm. To achieve this, you will have to apply some ingenuity in figuring out how to update the fingerprint in only constant time for most positions in the array.

*Hint:* We can view an $m \times m$ matrix as an $m^2$-bit integer. Rather than computing its fingerprint directly, compute instead a fingerprint for each row first, and maintain these fingerprints as you move around.