1. Let A be an unsorted array of n distinct integers. An element a stored in A is called an *approximate minimum* if the rank of a in A is at most $n/4$. We want to find an approximate minimum of A. We can find the smallest element of A in $O(n)$ time, and report it. We instead run the following $O(1)$-time Monte Carlo algorithm to find an approximate minimum of A.

> Pick a uniformly randomly from A.
> Return a.

(a) Find the error probability for this algorithm.
(b) How can you reduce the error probability by repeating the experiment a constant number of times? By how much?

2. Let A be an unsorted array of n distinct integers. An element a stored in A is called an *approximate median* of A if the rank of a in A is in the range $n/4$ to $3n/4$. The problem in this exercise is to find an approximate median of A. We know that the exact median of A can be found (and reported) in $O(n)$ time. We instead run the following $O(1)$-time Monte Carlo algorithm for solving the current problem.

> Pick a uniformly randomly from A.
> Return a.

(a) Find the error probability for this algorithm.
(b) How can you reduce the error probability by repeating the experiment a constant number of times? By how much?

3. Given three $n \times n$ real-valued matrices A, B, and C, we need to check if $AB = C$. The simple deterministic algorithm will take $O(n^3)$ time, which can be improved to about $O(n^{2.37})$ using the best-known matrix-multiplication algorithms. Design an $O(n^2)$-time Monte Carlo algorithm for the problem. Is the algorithm yes-biased or no-biased or with both-sided errors? Determine the error probability.

4. Let $A = (a_1, a_2, \ldots, a_n)$ and $B = (b_1, b_2, \ldots, b_n)$ be two unsorted arrays. The elements of the arrays are chosen from a large range of integers. Repetitions are allowed in each array. Your task is to find whether A is a permutation of B. This problem can be solved in $O(n \log n)$ time by sorting the two arrays. We instead plan to solve this problem in $O(n)$ time by a Monte Carlo algorithm using a random hash function h. You do not want the error probability to be more than 1/2. You do not have to design h, but assume that for any given s, the hash function h can produce a uniformly random integer in the range $0, 1, 2, \ldots, s - 1$. Propose a Monte Carlo algorithm to solve this problem, and show that it gives the desired error probability.

5. Suppose that at each step of the min-cut algorithm, instead of choosing a random edge for contraction, two vertices are chosen at random and are merged into a single vertex. This is repeated until only 2 vertices are left. Show that there exist inputs, for which the modified algorithm finds a min-cut with exponentially small probability.

6. Augment the Bloom filter data structure so that deletion is possible. What is the probability that an element not in the set is deleted?

7. (a) A browser company has a huge database of $n = 5{,}000{,}000$ malicious URLs to check against when a user enters a URL, but it takes a long time to do a database lookup as the database is stored centrally and is not in the client computer where the browser is running. The browser designer wants to have a quick check if the URL entered is malicious or not in the web browser itself. Show how you can do it efficiently using a Bloom filter. Calculate the space saved (over a naive implementation when all URLs are directly stored in the client computer running the browser) if each URL takes on the average around 40 bytes. Assume that $k = 5$ hash functions are used, and the filter has size $s = 40{,}000{,}000$ bits. What is the false positive probability in this case?

(b) In this part, we use the bloom filter of Part (a). Suppose that an average user attempts to visit 100,000 URLs in a year, only 2,000 of which are actually malicious. The user wants to check whether each of these 100,000 URLs is malicious before accessing the site. Suppose that it takes half a second for the browser to check the central database for each URL, and only 1 millisecond for the browser to check in the local bloom filter.

Compare the following two times to check all 100,000 URLs.

(i) The time (in seconds) taken if no bloom filter is used, that is, every URL check is done by communicating with the central database.

(ii) The expected time (in seconds) taken if a bloom filter + database combination is used. In this case, a central database lookup is made only if the local bloom filter reports a URL as malicious.

8. [*Coupon collector's problem*] The UVW Chocolate Company prepares large numbers of n different types of coupons, and inserts them in chocolate packets to be sold in a city. The company makes sure that it has distributed an equal number of coupons of each type. If a customer can produce all of the n types of coupons to the company, she will receive a sedan as a gift from the company. In order that the company does not end up gifting too many sedans, it adopts a trick. It distributes the coupons in such a way that people from any given locality experience scarcity of some coupon types. Ms. Randoma is determined to win a gift. She guesses that the company may have played some tricks with its customers. She makes a whole-day tour in the entire city, chooses shops at random locations, and buys random chocolate packets. Find the expected number of chocolate packets she should buy in order that all of the n types of coupons are available to her. Because Ms. Randoma picks chocolate packets randomly, assume that in each packet, each of the n types of coupons is equally likely to occur in each buy.

9. A thief steals $n > 2$ gold balls of identical shape and size. Later, he comes to know from an informant that exactly one of the balls is made of city gold. He also learns that the city-gold ball has weigth slightly different from a (genuine) gold ball. He wants to separate out the city-gold ball from his collection of n balls so that he can sell the $n - 1$ gold balls, and give the city-gold ball to his wife as a gift. However, he needs a precision instrument to detect slight weight variations. The Bar Foo Jewellery Shop has one such instrument which can only compare the weights of two balls. However, for each use of the instrument, they charge a hefty 1000 Rs fee. So the thief wants to minimize the number of weighings. To that effect, he runs the following Las Vegas algorithm.

> Randomly permute the balls as $b_1, b_2, b_3, \ldots, b_n$.
> Compare b_1 with b_2.
> If the weights are different, then:
>   Compare b_1 with b_3.
>   If the weights are again different, return b_1, else return b_2.
> Else:
>   For $i = 3, 4, 5, \ldots, n - 1$, repeat:
>   Compare b_1 with b_i.
>   If the weights are different, return b_i.
>   Return b_n.

Deduce the expected number of weighings done by this algorithm. How does randomization help here?

10. Consider the situation of the previous exercise. Suppose that the weighing machine of Bar Foo Jewellery Shop can compare any number of balls (equal numbers on both sides). Assume that n is a power of two. Prove that the thief can identify the city-gold ball using $\log_2 n$ weighings.

11. Consider again the situation of the previous two exercises. Suppose that the Bar Foo Jewellery Shop charges $1000k$ Rs if k balls are compared with k balls in one weighing. What is the payment that the thief needs to make if he uses the algorithm of the previous exercise? Can randomization help the algorithm?

12. [*Quick Select*] Let A be an unsorted array of n distinct integers, and k an integer in the range $1 \le k \le n$. Your task is to find the k-th smallest element of A. To that effect, you choose an element uniformly randomly from A. Using that element as the pivot, you partition the array. Suppose that the pivot occupies the position j in the sorted array (assume that array indexing is 1-based). If $j = k$, you return the pivot. Otherwise, you make a recursive call on the smaller or larger (than the pivot) part. Deduce the expected and worst-case running times of this algorithm.

13. A Las Vegas algorithm always outputs correct answers. Let us investigate a similar class of algorithms which may sometimes report *failure*. But whenever the algorithms succeed, the outputs are correct. Let us call such an algorithm a Las Vegas′ algorithm. Let A' be a Las Vegas′ algorithm for some problem. Using this algorithm, design a Las Vegas algorithm A that never outputs *failure*. Assume that A' outputs *failure* with probability $\leq 1/2$. Express the expected running time of A in terms of the expected running time of A'.

14. A randomized algorithm is called an *Atlantis City algorithm* if it is probably correct and probably fast. You are given a positive integer l. Your task is to locate a random l-bit prime number. Propose an Atlantis City algorithm for this problem. Deduce the error probability and the expected running time of your algorithm.

You may assume the *prime number theorem* which states that for a positive real number x, the number of primes $\leq x$ is approximately $x / \ln x$.

15. Propose an efficient algorithm for computing a random permutation of 1, 2, 3, …, n. You should ensure that your algorithm outputs a permutation with probability $1 / n!$.