1. Suppose we initialize an array $A[1..n]$ by setting $A[i] = i$ for all $i$, and then randomly shuffle the array. After shuffling, each of the $n!$ possible permutations of the array $A$ is equally likely. An index $i$ is called a ***fixed point*** of the shuffling permutation if $A[i] = i$.

   Let $X$ denote the number of fixed points of this random permutation.

   (a) What is the exact value of $E[X]$? *[Hint: Express $X$ as a sum of indicator variables.]*

   (b) What is the exact value of $E[X^2]$? *[Hint: What is the probability that two indices $i$ and $j$ are both fixed points?]*

   ***Prove*** that both of your answers are correct.

2. A *multilinear polynomial* is a sum of *terms*, where each term is a product of *distinct* variables and a non-zero real coefficient. The degree of a single term is the number of variables it contains. For example, the expression

$$x_2 - 5x_1x_4 + 6x_2x_3 - 8x_1x_2x_3x_4$$

   is a multilinear polynomial with three terms, which have degrees 1, 2, 2, and 4.

   Let $P$ be a multilinear polynomial with $n$ variables $x_1, \ldots, x_n$ and $m$ terms, where the degree of every term is exactly $n/10$. We call a set of variables $H \subseteq \{x_1, \ldots, x_n\}$ a ***hitting set*** for $P$ if $H$ contains at least one variable from each term of $P$.

   Suppose we randomly generate $H$ by independently including each variable $x_i$ with probability $p = (c \log m)/n$, for some constant $c \geq 1000$.

   (a) ***Prove*** that for any fixed term in $P$, the set $H$ contains at least one variable from that term with probability at least $1 - 1/m^3$. *[Hint: First compute the probability exactly, and then use the world's most useful inequality.]*

   (b) ***Prove*** that $H$ contains at most $c^2 \log m$ variables with probability at least $1 - 1/m^2$.

   (c) ***Prove*** that $H$ is a hitting set of size $O(\log m)$ with probability at least $1 - 1/m$.

   You may choose any constant value $c \geq 1000$ to make your calculations easier.

3. Suppose you have a set $X$ of $n$ items from some universe $\mathcal{U}$ that you want to store in a simple array $A[1..n]$, so that later you can look up elements of $X$ in worst-case constant time.

   Your manager really dislikes hash tables; randomized algorithms make them nervous. Instead, they suggest that you use five carefully engineered *access* functions $h_1, h_2, h_3, h_4, h_5$, each of which takes an element of $\mathcal{U}$ as input and returns an integer between 1 and $n$ as output, in constant time.

   These access functions are ***flawless*** if it is possible store each element $x \in X$ at one of the five addresses $A[h_1(x)]$, $A[h_2(x)]$, $A[h_3(x)]$, $A[h_4(x)]$, or $A[h_5(x)]$, with no collisions—each array entry $A[i]$ must store exactly one element of $X$.

   Describe and analyze an algorithm to determine whether the given access functions are flawless. The input to your algorithm is the set $X$ and the access functions $h_1, h_2, h_3, h_4, h_5$.

*Question 4 is on the other side of this page.*

4. Recall that a *3CNF formula* is a boolean formula in conjunctive normal form with three literals per clause. Specifically:

   - A *literal* is either a variable $x_i$ or its negation $\neg x_i$.
   - A *clause* is the disjunction (OR) of exactly three literals, using three distinct variables.
   - Finally, a 3CNF formula is the conjunction (AND) of one or more clauses.

   For example, the following expression is a 3CNF formula with four variables and four clauses:

   $$(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (x_2 \vee x_3 \vee \neg x_4).$$

   This formula evaluates to TRUE under the assignment

   $$x_1 = \text{FALSE},\ x_2 = \text{TRUE},\ x_3 = \text{TRUE},\ x_4 = \text{TRUE}.$$

   Suppose you are given a 3CNF formula with $n$ variables $x_1, \ldots, x_n$ and $m$ clauses, where $m \geq 100$.

   (a) Suppose we independently assign each variable $x_i$ to be TRUE or FALSE with equal probability. A clause is *satisfied* by this assignment if the clause evaluates to TRUE; otherwise, the clause is *unsatisfied*. What is the exact expected number of ***unsatisfied*** clauses under this assignment? *[Hint: Express the number of unsatisfied clauses in terms of indicator variables.]*

   (b) ***Prove*** that the probability that at least $m/8+1$ clauses are unsatisfied is at most $1-C/m$ for some constant $C$. *[Hint: You may use that $\frac{1}{1+t} \geq \frac{1}{2t}$ for all $t \geq 1$.]*

   (c) Part (b) implies that under a random assignment, the number of satisfied clauses is at least $7m/8$ with probability at least $C/m$. Using this fact, describe an *efficient* randomized algorithm that *always* finds an assignment that satisfies at least $7m/8$ clauses, and analyze its expected running time.