1. [15 pts] In a popular form of logic puzzles, you land on an island that has three types of inhabitants: “knights”, who always tell the truth; “knaves”, who always lie; and “spies”, who sometimes lie and sometimes tell the truth.

Suppose there are \( n \) inhabitants, where 60% are known to be decent folks, i.e., knights. The remaining 40% are bad, i.e., knaves or spies. You want to know who are the good/bad guys, i.e., determine the types of all \( n \) inhabitants. You are allowed to ask only questions of the form, “is person A a knight/knave/spy?”, to another person B. (All the inhabitants know each other.) Obviously, if you can find a person who you know is a knight, the problem is solved after asking \( n \) additional questions.

(a) [2 pts] Give a (very) efficient Monte-Carlo algorithm that finds a knight. State the probability of error. (Hint: this is supposed to be easy!)

(b) [13 pts] Give a Las-Vegas algorithm that finds a knight by asking \( O(n) \) expected number of questions. Analyze the constant factor in the big-Oh and make it smaller than 1.5. (Hint: use (a). How can you confirm whether a specific person is a knight by asking \( O(n) \) questions?)

(Note: there is also a deterministic algorithm that requires \( O(n) \) questions, but it is more complicated and has a larger constant.)

2. [18 pts] We are given two strings \( x \) and \( y \) of length \( n \) over the alphabet \{0,1\}.

(a) [16 pts] Given an integer \( k \), present an efficient Las Vegas algorithm to decide whether there exists a string of length \( k \) that is both a substring of \( x \) and a substring of \( y \) (and if so report the substring). Aim for linear time if possible.

(Hint: fingerprints...)

(b) [2 pts] Using part (a), show how to efficiently find the longest common substring of \( x \) and \( y \). (For example, if \( x = \text{ARITHMETIC} \) and \( y = \text{ALGORITHMS} \) over the English alphabet, the answer is \text{RITHM}.) Aim for \( O(n \log n) \) time.

3. [17 pts] We are given a set of \( n \) elements, where \( d \) of them are “defective” (\( d < n/2 \)). We want to identify the defective elements. The only allowed operation is the following test: given a subset \( S \), if \( S \) contains no defective elements, the tester reports “ok”; if \( S \) contains exactly one defective element, the tester reports this element; however, if \( S \) contains two or more defective elements, the tester reports “inconclusive”.

(a) [4 pts] Give a deterministic algorithm that can find one of the \( d \) defective elements with \( O(\log n) \) tests. (Note: this is the best possible, as there is a matching lower bound for deterministic algorithms.)
(b) [10 pts] Give a Las Vegas algorithm that can find one of the $d$ defective elements with $O(1)$ expected number of tests.

(Hint: pick a subset where each element is chosen with probability $1/d$. Show that the probability that exactly 1 defective element appears in the subset is $\Omega(1)$. It might be helpful to know that $\lim_{k \to \infty} (1 - 1/k)^k$ is a constant.)

(c) [3 pts] Give a Las Vegas algorithm that finds all defective elements with $O(d)$ expected number of tests (using (b)).