**Question 1.** *Suppose you have a biased coin that when flipped takes on heads with unknown probability $p$ and tails with probability $1-p$. Show how to use this coin to construct a string of $n$ independent bits such that each bit is equally likely to be a 0 or a 1. In other words, show how to use this coin to construct an algorithm that, when run, behaves likely an unbiased coin. What is the expected running time of your algorithm as a function of $p$?*

*Proof.* We know that the consecutive sequence HT has the same probability as the consecutive sequence TH. So we would sample the baised coin 2 at a time, if we get HH or TT, we would throw the result out. If we get HT, then HEADS, otherwise if we get TH, then TAILS. Since $\frac{1}{2p(1-p)}$ is the number of pairs we expect to sample before we get either HT or TH. So the expected number of sampling the baised coin is $\frac{1}{2p(1-p)} \times 2 = \frac{1}{p(1-p)}$. □

**Question 2.** *[Inspired by Dave Moore] Imagine a procedure* $\text{RANDOM}(a, b)$ *that, when called, returns an integer between $a$ and $b$ inclusively and uniformly at random. That is, each integer in the range $[a, b]$ is equally likely to appear on a call to* $\text{RANDOM}(a, b)$. *Now, suppose you have a fair coin. Describe an implementation of* $\text{RANDOM}(a, b)$ *that is only allowed to flip this coin (i.e., it can't use any other source of randomness). What is the expected running time of your procedure, as a function of $a$ and $b$?*

*Proof.* We will produce integers in the range $[0,\ b-a]$, and then map it into $[a, b]$. We will flip a coin $j = \lceil log_2(b-a) \rceil + 1$ times, and let the result of each coin represent a binary bit (e.g. heads means 0 and tails means 1). We convert the resulting binary number into its decimal represention, $d$. We know that $d$ is uniformly distributed in the interval $[0,\ 2^j - 1]$, so if $d > (b-a)$, then we rerun the algorithm until we can an integer in the range $[0,\ b-a]$. For each iteration we need the sample the coin $j$ times, and we expect to have $\frac{2^j}{b-a}$ iterations. So the expected running time is $\frac{j2^j}{b-a} = \frac{(\lceil log_2(b-a) \rceil + 1)2^{(\lceil log_2(b-a) \rceil + 1)}}{b-a} \leq \frac{(\lceil log_2(b-a) \rceil + 1)4(b-a)}{b-a} \approx O(log(b-a))$. □

**Question 3.** *Consider a very simple online auction system that works as follows. There are $n$ bidding agents; agent $i$ has a bid $b_i$, which is a positive natural number. We will assume that all bids $b_i$ are distinct from one another. The bidding agents appear in an order chosen uniformly at random, each proposes its bid $b_i$ in turn, and at all times the system maintains a variable $b^*$ equal to the highest bid seen so far (Initially $b^*$ is set to 0).*

*What is the expected number of times that $b^*$ is updated when this process is executed, as a function of the parameters in the problem?*

Example. *Suppose $b_1 = 20$, $b_2 = 25$, and $b_3 = 10$, and the bidders arrive in the order 1, 3, 2. Then $b^*$ is updated for 1 and 2, but not for 3.*

*Proof.* Let $X_i$ be an indicator variable that we will have an update on the $i$th bid. We will only have an update on the $i$th bid if and only if the $i$th bid is the best amongst the first $i$ bids. This happens with expectly probability $\frac{1}{i}$, since the best bid amongst the first $i$ has equal probability in each of the $i$ positions. So (expected number of updates) = $\sum_i^n X_i = \sum_{i=1}^n \frac{1}{i} \equiv O(log\, n)$. □

**Question 4.** *Consider a county in which 100,000 people vote in an election. There are only two candidates on the ballot: a Democratic candidate (denoted $D$) and a Republican candidate (denoted $R$). As it happens, this county is heavily Democratic, so 80,000 people go to the polls with the intention of voting for $D$, and 20,000 go to the polls with the intention of voting for $R$.*

*However, the layout of the ballot is a little confusing, so each voter, independently and with probability $1/100$, votes for the wrong candidate – that is, the one that he or she didn't intend to vote for. (Remember that in this election, there are only two candidates on the ballot.)*

*Let $X$ denote the random variable equal to the number of votes received by the Democratic candidate $D$, when the voting is conducted with this process of error. Determine the expected value of $X$, and give an explanation of your derivation of this value.*

*Proof.* For sake of argument we will call people intending to vote for $D$ democrats and people intending to vote for $R$ republicans. $X = 20000 * (1/100) + 80000 * (99/100) = 79400$ because we expect 1% of republicans to wrongly vote for $D$ and 99% of democrats to rightly vote for $D$. □

**Question 5.** *[extra credit] Suppose you have the same biased coin as in Question 1, however, this time you don't wish to construct an algorithm to produce an unbiased coin (i.e. a coin with probability 1/2 of coming up heads) but rather another unbiased coin with some given probability $q$ of coming up heads. Show how to use your coin to construct an algorithm that, when run, behaves like a biased coin with probability $q$ of returning heads. You may assume that $q$ is a rational number.* Note: I don't know if this problem has a solution. Let's see.

*Proof.* From question (1) we know that we can simulate a fair coin using a biased coin with probability p of coming up heads. Now we simply need to show that we can use a fair coin to simulate a biased coin with probability $q$ of returning heads. Since $q$ is a rational number, let $q = \frac{a}{b}$, where $a$ and $b$ are coprime. Using the result of question (2) we know that we can use a fair coin to pick a random integer $n$ uniformly distributed in the interval $[0, b]$. We can stipulate that if $n < a$, then HEADS, else TAILS, and we note that HEADS will occur with exactly probability $\frac{a}{b} = q$. The running time of this algorithm, combining the running time analysis of (1) and (2), is $\frac{O(\log b)}{p(1-p)}$. □