1. (a) Prove that the item returned by GetOneSample(S) is chosen uniformly at random from S.

Solution: Let $P(i,n)$ denote the probability that GetOneSample(S) returns the ith item in a given stream S of length n. We consider the behavior of the *last* iteration of the algorithm. With probability $1/n$, GetOneSample returns the last item in the stream, and with probability $(n-1)/n$, GetOneSample returns the recursively computed sample of the first $n-1$ elements of S. Thus, we have the following recurrence for all positive integers i and n:

$$P(i,n) = \begin{cases} 0 & \text{if } i > n \\ \dfrac{1}{n} & \text{if } i = n \\ \dfrac{n-1}{n} P(i,n-1) & \text{if } i < n \end{cases}$$

The recurrence includes the implicit base case $P(i,0) = 0$ for all i. The induction hypothesis implies that $P(i,n-1) = 1/(n-1)$ for all $i < n$. It follows immediately that $P(i,n) = 1/n$ for all $i \le n$, as required. ■

Rubric: 3 points.

(b) What is the *exact* expected number of times that GetOneSample(S) executes line ($\star$)?

Solution: Let X_i be the indicator variable that equals 1 if line ($\star$) is executed in the ith iteration of the main loop; then $X = \sum_i X_i$ is the total number of executions of line ($\star$). The algorithm immediately gives us $\mathrm{E}[X_i] = \Pr[X_i = 1] = 1/i$, so linearity of expectation implies $\mathrm{E}[X] = \sum_{i=1}^n \mathrm{E}[X_i] = \sum_{i=1}^n 1/i = \boldsymbol{H_n}$. ■

Rubric: 2 points.

(c) What is the *exact* expected value of ℓ when GetOneSample(S) executes line ($\star$) for the *last* time?

Solution: Let ℓ^* denote the value of ℓ when GetOneSample(S) executes line ($\star$) for the last time. Part (a) immediately implies that ℓ^* is uniformly distributed between 1 and n. It follows that $\mathbf{E}[\boldsymbol{\ell^*}] = \mathbf{(n+1)/2}$. ■

Rubric: 2 points.

(d) What is the *exact* expected value of ℓ when either GetOneSample(S) executes line ($\star$) for the *second* time (or the algorithm ends, whichever happens first)?

Solution: Let ℓ_2 denote the value of ℓ when either GetOneSample(S) executes line ($\star$) for the second time (or the algorithm ends, whichever happens first). In order to determine

$$\mathrm{E}[\ell_2] = \sum_{i=1}^n \Pr[\ell_2 \ge i],$$

we compute $\Pr[\ell_2 \ge i]$ for all i.

We immediately have $\Pr[\ell_2 \geq 1] = 1$. For any $i > 1$, we have $\ell_2 \geq i$ if and only if line $(\star)$ is *not* executed in iterations 2 through $i-1$ of the main loop, so

$$\Pr[\ell_2 \geq i] = \prod_{j=2}^{i-1} \frac{j-1}{j} = \frac{1}{i-1}.$$

We conclude that

$$\mathrm{E}[\ell_2] = \sum_{i=1}^{n} \Pr[\ell_2 \geq i] = 1 + \sum_{i=2}^{n} \frac{1}{i-1} = \boldsymbol{H_{n-1} + 1}.$$

■

Rubric: 3 points.

(e) Describe and analyze an algorithm that returns a subset of k distinct items chosen uniformly at random from a data stream of length at least k. The integer k is given as part of the input to your algorithm. Prove that your algorithm is correct.

Solution ($O(k)$ time per element): Perhaps the more intuitive solution is to chain together k instances of GetOneSample, where the items rejected by each instance are treated as the input stream for the next instance. Folding these k instances together gives us the following algorithm, which runs in ***$O(kn)$ time***:

```
GetSamples($S, k$):
  for $\ell \leftarrow 1$ to $\infty$
      if $S$ is done
          return $sample[1..k]$
      $x \leftarrow$ next item in $S$
      for $j \leftarrow 1$ to $\min\{k, \ell\}$
          if Random($\ell - j + 1$) $= 1$
              swap $x \leftrightarrow sample[j]$
```

Correctness follows inductively from the correctness of GetOneSample as follows. The analysis in part (a) implies that $sample[1]$ is equally likely to be any element of S. Let $S'[1..n-1]$ be the sequence of items rejected by $sample[1]$; at each iteration ℓ, the item $S'[\ell-1]$ is either $S[\ell]$ (with probability $1-1/\ell$) or the previous value of $sample[1]$ (with probability $1/\ell$). The output array $sample[2..k-1]$ has the same distribution as the output array from GetSamples($S', k-1$). Thus, the inductive hypothesis implies that $sample[2..k-1]$ contain $k-1$ distinct items chosen uniformly at random from $S \setminus \{sample[1]\}$. We conclude that $sample[1..k-1]$ contain k distinct items chosen uniformly at random from S, as required. ■

Solution ($O(1)$ time per element): To develop the algorithm, let's start with a standard algorithm for a different problem. Recall from the lecture notes that the ***Fisher-Yates algorithm*** randomly permutes an arbitrary array. (This algorithm is called InsertionShuffle in the notes.)

```
FisherYates($S[1..n]$):
  for $\ell \leftarrow 1$ to $n$
      $j \leftarrow$ Random($\ell$)
      swap $S[j] \leftrightarrow S[\ell]$
  return $S$
```

We can prove that each of the $n!$ possible permutations of the input array has the same probability of being output by FisherYates, by combining two observations:

- First, there are exactly $n!$ possible outcomes of all the calls to Random. Specifically, the ℓth call to Random has ℓ possible outcomes, so the total number of outcomes is exactly $\prod_{\ell=1}^{n} \ell = n!$.
- Second, every permutation of S can be computed by FisherYates. Specifically, the location of $S[n]$ in the output array gives us a unique outcome for the last call to Random, and (after undoing the last swap) the induction hypothesis implies that there is a sequence of Random outcomes that produce any desired permutation of $S[1..n-1]$. (The base case $n = 0$ is trivial.)

In particular, each subset of k distinct elements of S has the same probability of appearing in the prefix $S[1..k]$ of the output array.

Now suppose we modify the Fisher-Yates algorithm in two different ways. First, let the input be given in a stream rather than an explicit array; in particular, we do not know the value of n in advance. Second, we maintain only the first k elements of the output array.

```
FYSample(S, k):
    for $\ell \gets 1$ to $k$
        $sample[\ell] \gets$ next item in $S$
        swap $sample[\ell] \leftrightarrow sample[\text{Random}(\ell)]$
    for $\ell \gets k+1$ to $\infty$
        if $S$ is done
            return $sample[1..k]$
        $x \gets$ next item in $S$
        $j \gets \text{Random}(\ell)$
        if $j \le k$
            $x \gets sample[j]$
```

The k elements produced by this modified algorithm have *exactly* the same distribution as the first k elements of the output of Fisher-Yates. Thus, FYSample chooses a k-element subset of the stream uniformly at random, as required. (In fact, the algorithm is still correct even if we remove the swap in the initial for-loop.) The algorithm runs in $\boldsymbol{O(n)}$ ***time***, using only $O(k)$ space. ■

Rubric: Grade part (e) as a separate 10-point homework problem: 5 points for the algorithm + 5 points for the proof. These are neither the only correct algorithms nor the only correct proof for these algorithms.

2. Let $A[0\,..\,2^w-1]$ and $B[0\,..\,2^w-1]$ be arrays of independent random ℓ-bit strings, and define the hash function $h_{A,B}\colon \mathcal{U} \to [m]$ by setting

$$h_{A,B}(x,y) := A[x] \oplus B[y]$$

where $\oplus$ denotes bit-wise exclusive-or. Let $\mathcal{H}$ denote the set of all possible functions $h_{A,B}$. Filling the arrays A and B with independent random bits is equivalent to choosing a hash function $h_{A,B} \in \mathcal{H}$ uniformly at random.

(a) Prove that $\mathcal{H}$ is 2-uniform.

Solution: Let (x,y) and (x',y') be arbitrary distinct elements of $\mathcal{U}$, and let i and j be arbitrary (possibly equal) hash values. To simplify notation, we define

$$a = A[x], \quad b = B[y], \quad a' = A[x'], \quad \text{and} \quad b' = B[y'].$$

Say that a, b, a', b' are ***good*** if $a \oplus b = i$ and $a' \oplus b' = j$. We need to prove that

$$\Pr\big[a, b, a', b' \text{ are good}\big] = \frac{1}{m^2}.$$

There are three cases to consider.

- Suppose $x \neq x'$ and $y \neq y'$. Then a, b, a', b' are four different and therefore independent random w-bit strings. There are m^4 possible values for a, b, a', b'. If we fix a and a' arbitrarily, there is exactly one good value of b and exactly one good value of b', namely, $b = a \oplus i$ and $b' = a' \oplus j$. Thus, there are m^2 good values for a, b, a', b'. We conclude that the probability that a, b, a', b' are good is $m^2/m^4 = 1/m^2$.
- Suppose $x = x'$ and $y \neq y'$. Then $a = a'$, so there are only m^3 possible values for a, b, a', b'. If we fix $a = a'$ arbitrarily, there is exactly one good value of b and exactly one good value of b', namely, $b = a \oplus i$ and $b' = a' \oplus j$. Thus, there are m ***good*** values of a, b, a', b'. We conclude that the probability that a, b, a', b' are good is $m/m^3 = 1/m^2$.
- The final case $x \neq x'$ and $y = y'$ is symmetric with the previous case. ■

Rubric: 3 points. This is more detail than necessary for full credit. This is not the only correct solution. "See part (b)" is worth exactly the same number of points as your solution to part (b).

(b) Prove that $\mathcal{H}$ is 3-uniform. *[Hint: Solve part (a) first.]*

Solution: Let (x,y), (x',y'), (x'',y'') be arbitrary distinct elements of $\mathcal{U}$, and let i, j, k be arbitrary (possibly equal) hash values. To simplify notation, we define

$$a = A[x], \quad b = B[y], \quad a' = A[x'], \quad b' = B[y'], \quad a'' = A[x''], \quad b'' = B[y''].$$

Say that a, b, a', b', a'', b'' are ***good*** if $a \oplus b = i$ and $a' \oplus b' = j$ and $a'' \oplus b'' = k$. There are three cases to consider.

- Suppose x, x', x'' are all different. Arbitrarily fix y, y', y''. There are m^3 possible values for x, x', x'', but only one good value: $x = y \oplus i$ and $x' = y' \oplus j$ and $x'' = y'' \oplus k$.

- If $x = x' = x''$, then y, y', y'' must be all different, and we can argue exactly as in the previous case.
- The only remaining case (up to symmetry) is $x = x' \neq x''$ and $y \neq y' = y''$. Then there are m^4 possible values for a, b, b', a''. If we fix a arbitrarily, the only good values of the remaining variables are $b = a \oplus i$ and $b' = a \oplus j$ and $a'' = b' \oplus k = a \oplus j \oplus k$. Thus, there are exactly m good values for a, b, b', a''.

In all cases, we conclude that $\Pr[a, b, a', b', a'', b'' \text{ are good}] = 1/m^3$. ■

Rubric: 4 points. This is not the only correct solution.

(c) Prove that $\mathscr{H}$ is ***not*** 4-uniform.

Solution: For any function $h \in \mathscr{H}$ and any w-bit strings x, y, x', y', we have

$$\begin{aligned} h(x,y) \;\oplus\; h(x',y) \;\oplus\; h(x,y') \;\oplus\; h(x',y') \\ &= A[x] \oplus B[y] \;\oplus\; A[x'] \oplus B[y] \;\oplus\; A[x] \oplus B[y'] \;\oplus\; A[x'] \oplus B[y'] \\ &= A[x] \oplus A[x] \;\oplus\; A[x'] \oplus A[x'] \;\oplus\; B[y] \oplus B[y] \;\oplus\; B[y'] \oplus B[y'] \\ &= 0. \end{aligned}$$

It follows that for any hash values i, j, k, l, the probability

$$\Pr\big[h(x,y) = i \;\wedge\; h(x,y') = j \;\wedge\; h(x',y) = k \;\wedge\; h(x',y') = l\big]$$

is an integer multiple of $1/m^3$, and therefore cannot equal $1/m^4$. ■

Rubric: 3 points. This is more detail than necessary for full credit. This is not the only correct solution.