1. In this problem we consider yet another method for universal hashing. Suppose we are hashing from the universe $U = \{0, 1, \ldots, 2^w - 1\}$ of $w$-bit strings to a hash table of size $m = 2^\ell$; that is, we are hashing $w$-bit words into $\ell$-bit labels. To define our universal family of hash functions, we think of words and labels as boolean vectors of length $w$ and $\ell$, respectively, and we specify our hash function by choosing a random boolean matrix.

For any $\ell \times w$ matrix $M$ of 0s and 1s, define the hash function $h_M : \{0, 1\}^w \to \{0, 1\}^\ell$ by the boolean matrix-vector product

$$h_M(x) = Mx \mod 2 = \bigoplus_{i=1}^w M_i x_i = \bigoplus_{i : x_i = 1} M_i,$$

where $\oplus$ denotes bitwise exclusive-or (that is, addition mod 2), $M_i$ denotes the $i$th column of $M$, and $x_i$ denotes the $i$th bit of $x$. Let $\mathcal{M} = \{h_m \mid M \in \{0, 1\}^{w \times \ell}\}$ denote the set of all such random-matrix hash functions.

For example, suppose $w = 8$ and $\ell = 4$. Let $M$ be the $w \times \ell$ matrix

$$M = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

Then we can compute $h_M(173) = 12$ as follows:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \oplus \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

(a) Prove that $\mathcal{M}$ is a universal family of hash functions.

(b) Prove that $\mathcal{M}$ is not uniform.

(c) Now consider a modification of the previous scheme, where we specify a hash function by a random matrix $M \in \{0, 1\}^{\ell \times w}$ and an independent random offset vector $b \in \{0, 1\}^\ell$:

$$h_{M,b}(x) = (Mx + b) \mod 2 = \bigoplus_{i=1}^w M_i x_i \oplus b$$

Prove that the family $\mathcal{M}^+$ of all such functions is strongly universal (2-uniform).

(d) Prove that $\mathcal{M}^+$ is not 4-uniform.

(e) [Extra credit] Prove that $\mathcal{M}^+$ is actually 3-uniform.
2. **Reservoir sampling** is a method for choosing an item uniformly at random from an arbitrarily long stream of data.

```plaintext
GETONESAMPLE(stream S):
    ℓ ← 0
    while S is not done
        x ← next item in S
        ℓ ← ℓ + 1
        if RANDOM(ℓ) = 1
            sample ← x (⋆)
    return sample
```

At the end of the algorithm, the variable ℓ stores the length of the input stream S; this number is not known to the algorithm in advance. If S is empty, the output of the algorithm is (correctly!) undefined.

In the following questions, consider an arbitrary non-empty input stream S, and let n denote the (unknown) length of S.

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

(b) What is the exact expected number of times that GETONESAMPLE(S) executes line (⋆)?

(c) What is the exact expected value of ℓ when GETONESAMPLE(S) executes line (⋆) for the last time?

(d) What is the exact expected value of ℓ when either GETONESAMPLE(S) executes line (⋆) for the second time (or the algorithm ends, whichever happens first)?

3. (This is a continuation of the previous problem.) 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. Prove that your algorithm is correct. Your algorithm should have the following form:

```plaintext
GETSAMPLE(stream S, k):
    ⟨⟨Do some preprocessing⟩⟩
    while S is not done
        x ← next item in S
        ⟨⟨Do something with x⟩⟩
    return ⟨⟨something⟩⟩
```

Both the time for each ⟨⟨step⟩⟩ in your algorithm and the space for any necessary data structures must be bounded by functions of k, not the length of the stream.

For example, if k = 2 and the stream contains the sequence {♦, ♥, ♦, ♣}, your algorithm should return the subset {♦, ♣} with probability 1/6.