Problem 1. In this problem, we consider two-write WOM codes that their second write does not necessarily always succeed. Every such a WOM code is characterized by three parameters $r_1, r_2, n$, where $r_1 < n$ and $r_1 \leq n - r_2$, and a binary matrix $H$ of size $r_1 \times n$, which is chosen uniformly at random. The code is denoted by $\mathcal{C}(n, r_1, r_2, H)$. On the first write, one of $\sum_{i=0}^{n-r_2} \binom{n}{i}$ messages is written such that at most $n - r_2$ cells are programmed from zero to one. Let $c_1$ be the memory-state vector after the first write. On the second write, the user seeks to write a message $s$ of $r_1$ bits by choosing a vector $c_2$ such that $H \cdot c_2^T = s^T$ and $c_2 \geq c_1$. Assume the messages on the first and second write are chosen uniformly at random, we define by $P(n, r_1, r_2, H)$ the probability that the second write succeeds. Prove that for all $p \in (0, 0.5)$ and $\epsilon > 0$ there exists a WOM code $\mathcal{C}(n, r_1, r_2, H)$ such that its rates $R_1, R_2$ and success probability $P(n, r_1, r_2, H)$ satisfy the following requirement:

$$R_1 \geq h(p) - \epsilon, \quad R_2 \geq 1 - p - \epsilon, \quad P(n, r_1, r_2, H) \geq 1 - \epsilon.$$ 

Problem 2. In this problem we will design a non-binary two-write WOM codes. Assume there are $n q$-ary cells, where $q$ is a multiple of 3 and there exists a binary two-write WOM code $[n, 2^{nR_1}, 2^{nR_2}]$.

(a) Design a two-write $q$-ary $[n, 2^{nR_1 \cdot (q/3)^n}, 2^{nR_2 \cdot (q/3)^n}]$ WOM code and prove its correctness.

(b) Find the best sum-rate which is possible to achieve by this construction and compare it with the capacity upper bound on the sum-rate in this case.

Problem 3. In this problem we will study WOM codes with a buffer. Assume the user writes $t$ messages $M_1, \ldots, M_t$, where $t$ is even. After writing the $i$-th message, for $2 \leq i \leq t$, the decoder needs to be able to recover the messages $M_i$ and $M_{i-1}$.

(a) Show that it possible to achieve sum-rate approaching $\log(t/2 + 1)$.

(b) Show that $\log(t/2 + 1)$ is an upper bound on the sum-rate.
(c) Now assume that the user writes three messages $M_1, M_2, M_3$. After the first write, the first message $M_1$ should be recovered. After the second write the first two messages $M_1, M_2$ should be recovered. However, on the third write the user specifies which of the first two messages should be recovered together with $M_3$.

(i) Prove that the sum-rate in this setup is upper bounded by 1.5.

(ii) Design a code construction with the highest possible sum-rate you can find.

**Problem 4.** Your goal in this problem is to design a memory with fast reading according to the following assumptions and requirements:

1. Assume there are $n$ cells, each with $q$ levels $\{0, 1, \ldots, q-1\}$. Then, $q-1$ binary pages (messages) are stored into these cells $p_1, p_2, \ldots, p_{q-1}$, where every page should store the same number of bits, denoted by $k$.

2. The pages are received together and are encoded to the memory by an encoding function $E : (\{0, 1\}^k)^{q-1} \rightarrow \{0, \ldots, q-1\}^n$.

3. In order to know the level of each cell there are $q-1$ thresholds, between levels 0 and 1, levels 1 and 2, and so on until levels $q-2$ and $q-1$. When reading the memory cells with the threshold between levels $i$ and $i+1$, a binary length-$n$ vector $v$ is received such that $v_j = 1$ if and only if the value of the $j$th cell is greater than $i$.

4. The memory needs to efficiently accommodate reading requests of pages such that every page is read by applying exactly a single threshold. More explicitly, page 0 ($p_0$) is read by applying the threshold between levels $q-2$ and $q-1$, page 1 ($p_1$) is read by applying the threshold between levels $q-3$ and $q-2$, and so on page $q-1$ ($p_{q-1}$) is read by applying the threshold between levels 0 and 1.

Such a coding scheme will be called a *Fast Reading Code (FR Code)* and will be denoted by an $(n, q, k)$ FR code, where $n$ is the number of cells, $q$ is the number of levels in each cell, and $k$ is the number of bits in each page. In this question a binary WOM code which stores $t$ messages, each of the same number of bits $k$ will be denoted by an $[n, t, k]$ WOM code.

(a) Show how to construct a $(3, 3, 2)$ FR code. Hint: use the Rivest-Shamir $[3, 2, 2]$ WOM code.

(b) Prove that if there exists an $[n, t = q - 1, k]$ WOM code, then there exists an $(n, q, k)$ FR code.

(c) Explain in words what the difference between WOM codes and FR codes is, and why FR codes do not imply WOM codes.

(d) Prove that there exists a $(7, 5, 3)$ FR code. Hint: use the coset coding scheme with the length-7 Hamming code.

(e) Prove that for all $m \geq 3$ there exists a $(2^m - 1, 2^m - 2 + 3, m)$ FR code.

(f) **Bonus:** Prove that there exists a $(15, 9, 4)$ FR code.
Problem 5. The goal in this problem is to understand the connection between the write amplification (or erasure factor) and over-provisioning. You can use all the results and assumptions we derived in class. Assume there are $U$ logical pages and $T$ physical pages, so the over-provisioning is $\rho = (T - U)/U$ and the storage rate is $\alpha = U/T$. You can also assume that the number of pages in a block, $Z$, is large.

The $U$ logical pages are distributed into two groups of hot and cold pages, where the number of hot, cold pages is $H = fU, C = (1 - f)U$ for some $0 \leq f \leq 1$, respectively. On each page write, one of the hot, cold pages is written with probability $p, 1 - p$, for some $0.5 \leq p \leq 1$, respectively. Within these two groups, pages are written uniformly at random.

(a) Find the optimal partition of the memory into two parts to write the hot and cold pages separately such that the write amplification is minimized. Solve this problem for the greedy and the least recently used (LRU) garbage collection policies. Write the value of the write amplification you received in each case.

(b) Repeat the task in (a) when the naive scheme of two-write WOM codes is used, and the goal is to minimize the number of block erasures. Solve this for the two garbage collection policies.

You can present your solutions by an equation for the constraint and not an explicit value.