Solve the following problems before the exercise session and be prepared to present your solutions at the session.

1. The $\gamma$ and Golomb-Rice codes we discussed in Lecture 3 were suitable for encoding any integer $n > 0$. Write down the Elias $\gamma$ codes for the numbers 1, 6, 8, 17, 22. Write down the Golomb-Rice codes for $M = 4$ for the same numbers. How would you modify the coding procedures so that you are also able to encode the number 0?

2. Recall the Simple-9 word-aligned binary code from Lecture 4, which is designed for 32-bit words. Design a simple word-aligned binary code that works for 64-bit words. Specify a table of selectors that shows i) the number of codes per word; ii) the number of bits per code; and iii) the number of wasted bits per word for each selector.

3. We saw the following formula in class for generating the Golomb-Rice code for an integer $n > 1$:

$$Golomb-Rice(n) = \text{unary}(q(n) + 1) \cdot \text{binary}_{\log M} r(n),$$

where $q(n) = \lfloor (n - 1)/M \rfloor$, $r(n) = (n - 1) \mod M$, and ‘.’ denotes concatenation.

One can think of Golomb-Rice codes as assigning the set of positive integers $> 1$ to groups, each of size $M$. For example, for $M = 8$: the numbers 1..8 get assigned to group 0, and get the shortest codes; the numbers 9..16 go to group 1 and get the second shortest set of codes, et c.

Imagine a variant of Golomb codes (called, say, \textit{Exponential Golomb codes}) that instead of dividing the code space evenly into $n/M$ groups each of $M$ code words, divides the code space into $\log n$ code groups of size 1, 2, 4, 8, ... et c. Write down the codes for the first dozen or so codes.

What other code from class does this code remind you of?

4. Write pseudocode for encoding and decoding positive integers to and from variable byte codes. You can use the examples in the lecture notes to help check your solution.