Optional Graded Assignment

- This Assignment allows to add a bonus grade from 0 to 3 points (exceptionally 4 if outstanding) to the first part of the written exam valid within one year from the assignment and not cumulative with earlier bonuses
- The assignment consists of an implementation of the Peterson–Gorenstein–Zierler decoding algorithm based on the Matlab App Designer for binary characteristic primitive or non-primitive Reed-Solomon codes (refer to the notes and to the Blahut book)

$$g(x) = \prod_{i=1}^{2t} (x - \omega^{j_0+i-1}) \qquad \omega^n = 1$$

$$\omega = \alpha^3 \quad \alpha \in \mathbb{F}_{16} \qquad \omega^5 = 1$$

- It is strictly forbidden the use of
  - any Matlab code from the literature or Matlab internal functions except those explicitly allowed
  - chat-gpt or equivalent AI tool
  - interaction among different groups leading to very similar code
- Groups of up to 4 students are allowed; the bonus is common to all students in the group; the groups must be declared within two days from the assignment by e-mail from institutional account to giorgio.taricco@polito.it with all group members in CC

Optional Graded Assignment

- The Assignment must be delivered within the midnight of three weeks after the date of the assignment
- The finite field operations can be implemented by using the Matlab finite field function gf
- The evaluation is not subject to negotiation except in the case of manifest misunderstandings
- The evaluation considers the following characteristics:
  - The code must not crash
  - The code must work properly with any input
  - The code must check the input correctness
  - The code must be user friendly
  - The code should be comprehensive
  - The visual presentation should be satisfactory
- The Assignment must be delivered in two parts:
  1. A Matlab mlapp file
  2. Comprehensive pdf report prepared in LaTeX
- It must be uploaded in the “Elaborati” section of your PoliTo web page, not by email

Optional Graded Assignment

- The Matlab code must implement the following operations:
  - Input parameters:
    - q, d characterizing the primitive or non-primitive RS code with systematic encoding
    - N. of errors ν
    - Random generator initialization seed
  - Generate a random codeword according to the specifications and output it
  - Generate and output a random error $e(x)$
  - Calculate the syndromes and output them
  - Implement the steps of the decoding algorithm with suitable output
  - Display the decoding outcome (immediate decoding when all syndromes are zero, decoding through the algorithm, or decoding failure when the syndromes are nonzero, but the decoding algorithm fails)
  - Output the finite field polynomial words in (at least) the following user-selectable formats $[c(x) = \alpha x^5 + \alpha^2 x^3 + x \text{ in } \mathbb{F}_8]$:
    - $\alpha x^5 + \alpha^2 x^3 + x$
    - $(0, \alpha, 0, \alpha^2, 0, 1, 0)$
    - $(000, 010, 000, 100, 000, 001, 000)$

n for non-primitive n|q-1

c(x) → e(x) v(x) = c(x) + e(x)

αx^5+α^2x^3+x

α^3 = α+1 = 0·α^2+1·α+1

(011)