Problem 28: Secret sharing implementation

This problem is to implement Shamir’s secret splitting scheme. You should write three programs:

**dealer** takes three command line arguments: a secret s, a threshold τ, and a number of shares k, where 1 ≤ τ ≤ k. It writes 2k + 3 whitespace-separated decimal integers (with no labels) to standard output: a prime p, the numbers τ and k, and a list of k shares (1, s₁), ..., (k, sₖ), where the shares are computed from the secret s according to Shamir’s (τ, k) secret splitting scheme. In particular, dealer finds a suitable prime p, generates a random polynomial p(x) with coefficients in \( \mathbb{Z}_p \) that encodes the secret s, and then generates the k shares.

**filter** reads 2k + 3 numbers from standard input as written by dealer. It selects a random subset of τ distinct shares from among the k input shares and writes 2τ + 2 whitespace-separated decimal integers to standard output: a prime p, a number τ, and a list of the τ randomly-selected shares (i₁, sᵢ₁), ..., (iₜ, sᵢₜ).

**recover** reads 2τ + 2 numbers from standard input as written by filter. It finds the secret s determined from its inputs according to Shamir’s scheme and writes it to standard output.

You may assume that all numbers are less than 2³¹, so your program can use ordinary C integers rather than bother with the big number packages. However, since you need to generate a prime p, you might still find it convenient to use one of the primality-testing routines from those packages.

Problem 29: Coin-flipping

Do problem 13.3.2 in the textbook[1] which refers to the coin-flipping protocol of section 13.1.

Problem 30: Indistinguishability

We say that judge \( J(z) \) \( ε \)-distinguishes random variables X and Y if

\[ |\text{prob}[J(X) = 1] - \text{prob}[J(Y) = 1]| \geq ε. \]

Let \( U_n \) be the uniform distribution on binary strings of length \( n \). Let \( X_n \) be the distribution that results from \( n \) flips of a biased coin, where the probability of 1 (“heads”) is 2/3 and the probability of 0 (“tails”) is 1/3.

(a) What is the largest value of \( ε \) for which there exists a probabilistic polynomial time judge \( J(z) \) to \( ε \)-distinguish \( U_1 \) from \( X_1 \)? Describe such a judge.

(b) How large can \( ε \) be as a function of \( n \) for a judge that distinguishes \( U_n \) from \( X_n \)? Describe a judge achieving this level of distinguishability.

---