# **Pre-Coding Essentials (Component: crates/vm\_algo/src/tabulation/plurality.rs, Version/FormulaID: VM-ENGINE v0) — 37/89**

## **1) Goal & Success**

Goal: Deterministically compute **UnitScores** for **plurality** ballots from per-option vote counts and turnout.

Success: Returns exact integer scores per OptionId, preserves canonical option order, and carries turnout (ballots\_cast, invalid\_or\_blank, valid\_ballots). No floats, no RNG.

## **2) Scope**

In scope: Per-unit plurality tabulation; validation of non-negative counts; optional invariants (sum of option votes ≤ valid\_ballots).

Out of scope: Allocation, gates/thresholds, aggregation, tie resolution, I/O/schema.

## **3) Inputs → Outputs**

Inputs:

unit\_id: UnitId

votes: &BTreeMap<OptionId, u64> (raw counts per option)

turnout: Turnout (ballots\_cast, invalid\_or\_blank, valid\_ballots)

options: &[OptionItem] (to enforce canonical (order\_index, id) ordering)

Output:

UnitScores { unit\_id, turnout, scores: BTreeMap<OptionId, u64> } (scores sorted by canonical option order)

## **4) Entities/Tables (minimal)**

## **5) Variables (only ones used here)**

## **6) Functions (signatures only)**

rust

CopyEdit

use std::collections::BTreeMap;

use vm\_core::{

ids::{UnitId, OptionId},

entities::{Turnout, OptionItem},

};

/// Deterministic plurality tabulation.

pub fn tabulate\_plurality(

unit\_id: UnitId,

votes: &BTreeMap<OptionId, u64>,

turnout: Turnout,

options: &[OptionItem],

) -> UnitScores;

/// Internal: build canonical score map from provided votes and option list.

fn canonicalize\_scores(

votes: &BTreeMap<OptionId, u64>,

options: &[OptionItem],

) -> BTreeMap<OptionId, u64>;

/// Internal checks (enabled in debug; return Result in release if preferred).

fn check\_tally\_sanity(

votes\_sum: u64,

turnout: &Turnout,

) -> Result<(), TabError>;

## **7) Algorithm Outline (implementation plan)**

**Canonical order**

Iterate options in (order\_index, OptionId) order and pull votes.get(&opt.id).copied().unwrap\_or(0).

Insert into a new BTreeMap<OptionId,u64> to have deterministic iteration for downstream.

**Sanity checks**

All counts are non-negative (u64 already).

valid\_ballots = ballots\_cast - invalid\_or\_blank (trust Turnout constructor).

sum(scores.values()) ≤ valid\_ballots (not enforced for approval/score; **required** for plurality). If violated, return TabError::TallyExceedsValid.

**Assemble result**

Return UnitScores{ unit\_id, turnout, scores }.

**No normalization**

Do not divide or compute shares here; seats/gates use integers or ratios later.

## **8) State Flow**

Pipeline: **TABULATE** (this function) → **ALLOCATE** (WTA/PR) → **AGGREGATE** → **GATES**. UnitScores feed allocation and later gates/labels.

## **9) Determinism & Numeric Rules**

Determinism via canonical option iteration and BTreeMap storage.

Integer math only; no rounding; no RNG.

## **10) Edge Cases & Failure Policy**

Missing option in votes ⇒ treated as 0.

Extra option present in votes but not in options ⇒ ignore or error? **Choose error** (TabError::UnknownOption) to keep referential integrity; loader should prevent this earlier.

votes\_sum > valid\_ballots ⇒ **error** (TabError::TallyExceedsValid).

turnout.valid\_ballots == 0 ⇒ still return zeros; downstream gates handle legitimacy.

## **11) Test Checklist (must pass)**

Happy path: A/B/C/D votes (e.g., 10/20/30/40), turnout 100/0/100 → scores equal input; canonical order matches options.

Missing option key in votes yields 0, not panic.

Unknown option key in votes triggers TabError::UnknownOption.

Tally sanity: sum(votes) == valid\_ballots passes; sum(votes) > valid\_ballots fails.

Determinism: shuffle insertion order of votes → identical UnitScores.scores order and bytes when serialized canonically.