Student ID: _____ Name: _____ TA: _____

Rust Lab 10 — GUI in Rust (Iced)

10/9/2025

Goal: Build a basic calculator desktop app using the Iced GUI framework in Rust. You will design the state and message flow, assemble the UI, and implement the update logic yourself. This handout intentionally contains no solution code.

Learning Outcomes

- Apply the Iced Application architecture: State → View → Message → Update.
- Design minimal, testable state for a GUI app (display text, stored operand, pending operation, flags).
- Lay out a grid of widgets, align text, and wire user interactions to messages.
- Handle edge cases (e.g., chaining operations, divide-by-zero) and define acceptance tests.

Prerequisites

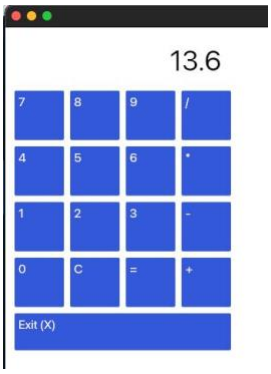
- Completed reading the Iced Beginner Handout.
- Rust toolchain installed (rustup + cargo).
- Comfort with building and running a Cargo project.

What You Must Deliver

- A compiling Rust project that runs a working calculator GUI.
- A short README.md explaining your state design and message flow (max 1 page).
- Screenshots demonstrating all required behaviors (see Test Checklist).

Functional Requirements

1. Buttons: digits 0–9, operators + − × ÷, =, C (clear), Exit.
2. Display: shows the current number; right-aligned, large font.
3. Number entry: digits append to the current number; after operator or '=', the next digit starts a new number.
4. Operators: pressing +/−/×/÷ stores or computes and then remembers the new operator (chaining).
5. Equals: computes the result using the stored left operand, current display, and pending op.
6. Clear: resets the calculator to an initial state (display '0', no pending op).
7. Exit: closes the program (simple behavior acceptable for this lab).
8. Divide-by-zero behavior must be defined (you choose: e.g., 'Infinity', error message, or disabled '=').



UI Layout Specification

Use a grid layout with uniform square buttons. The display row is right-aligned.

Reference layout (you must implement visually similar behavior)

Test Checklist (attach screenshots)

Input Sequence	Expected Display	Notes / Screenshot
2 + 3 =	5	
5 + 3 × 2 =	16	Chaining rule
9 ÷ 4 =	2.25 or policy	Define your rounding/policy
7 ÷ 0 =	Your chosen behavior	No crash
0 0 3	3	No leading zeros
1 2 C 4 =	4	Clear resets correctly
3 + = = =	9 (if repeat equals) or your policy	Document behavior
999999 + 1 =	1000000	Large numbers
Exit	App closes	

TA Checking: _____ Time: _____