Frontend

Desarrollar un componente reutilizable llamado **Number Input**, que pueda recibir sólo números en el cuadro de input (teclado) o bien incrementarse con los botones (mouse)

Además, debe cumplir con las siguientes funcionalidades:

- Solo puede recibir números. No debe mostrar texto en ningún caso (ignorarlo).
- Recibir props **value** y **onChange** para ser controlado desde afuera.
- El valor no puede ser comunicado hacia afuera de inmediato mientras el usuario escribe. Se debe implementar un *debounce* de 800ms con el objetivo de esperar a que el usuario termine de escribir.
- Recibir props **min** y **max** opcionales para limitar el valor que se puede ingresar.
    - Al alcanzar un límite, el botón correspondiente se debe deshabilitar.
    - Al ingresar un valor fuera del rango permitido, se debe sobreescribir el valor para cumplir con el máximo.
- Recibir props **softMin** y **softMax** opcionales, para advertir al usuario que está incumpliendo ciertos límites blandos. A diferencia del caso anterior, únicamente se pondrá de color rojo el cuadro de input y el valor en su interior.
- Recibir una prop **step** opcional, que indicará cuánto se debe disminuir/aumentar en cada click de los botones. El valor por defecto es 1.
- Recibir una prop **editable** opcional, que indicará si el valor es editable o no.
- Recibir una prop **precision=2** opcional, que indicará la cantidad de dígitos decimales a los que debe ser redondeado el valor antes de ser comunicado hacia afuera.
- Estilizado básico que permita distinguir cuándo estoy dentro del input (focus), cuando estoy sobre el botón (hover) y cuando estoy presionando el botón (active)
- BONUS: Implementar funcionalidad de mantener presionado el botón para incrementar rápidamente el valor. Además puede recibir una prop **longPressInterval** opcional, que indique la cantidad de milisegundos que se debe esperar para volver a incrementar el valor, cuando se mantiene presionado alguno de los botones.

Lógica

En el lenguaje de programación de su elección, implementar una función llamada **compute25()**, que reciba 4 dígitos como input (del 1 al 9, permitiendo repetidos) y entregue una expresión aritmética que al ser evaluada dé como resultado 25.
Si no existe una expresión, debe retornar "`SIN SOLUCIÓN`"

Debe cumplir las siguientes reglas:

- Solo se permiten las operaciones de suma, resta, multiplicación y división.
- No hay restricciones sobre cuáles y cuántas operaciones utilizar, entre las mencionadas.
- Se pueden usar paréntesis para agrupar las operaciones
- La división debe ser decimal (no se puede ignorar el resto)
- Se debe usar obligatoriamente una única vez cada uno de los 4 dígitos recibidos.
- No importa el orden en que se reciben los dígitos en el input. Pueden ser usados en cualquier orden en el output.
- Cada dígito debe ser usado por sí solo. No se pueden formar números más grandes concatenando los dígitos. Por ejemplo entregar 13 + 12 cuando se recibe 1,3,1,2 es ilegal.

En caso de existir más de 1 solución, se puede entregar cualquiera de ellas.
Se entregará una lista de 100 casos de prueba con su respectivo output.

Ejemplos:

Input	**Output**
compute25("8251")	1 + (8 * (5 - 2))
compute25("6153")	"SIN SOLUCIÓN"
compute25("7583")	(7 + 8) * (5 / 3)

SQL

Contexto:
Las partidas (*activity*) son trabajos llevados a cabo por contratistas o “maestros”. Por ejemplo: instalación de puertas, pintado, etc.

Los presupuestos (*budget*) representan 1 casa, estos pueden tener múltiples partidas cubicadas, las que a su vez pueden requerir múltiples materiales.
Nota: Los materiales se pueden repetir dentro de un mismo presupuesto en diferentes partidas. Por ejemplo, los tornillos o clavos se ocupan para muchas cosas.

La gerencia libera una determinada cantidad de material por cada partida y presupuesto. Con esto buscan que tanto las compras como la obra avancen de manera gradual.

Se realizan órdenes de compra, que piden ciertas cantidades de distintos materiales.

Los contratistas piden materiales a través de tickets para realizar una única partida a través de múltiples casas. (Por ejemplo, un día llegan y pintan 10 casas, por lo que piden latas de pintura para esos 10 budgets)

En casos excepcionales, los contratistas pueden pedir más material que el presupuestado. Este caso se llama sobrecarga y es material adicional que debe ser repuesto en las siguientes compras.

Objetivo:
Dadas las tablas mencionadas más adelante, usted debe calcular la cantidad a comprar, de cada material, en la siguiente Orden de Compra. Se debe tener en cuenta la cantidad liberada por la gerencia, la cantidad solicitada por tickets y la cantidad ya comprada con anterioridad. (Pensar en generar un balance)

Ejemplo:
Se tienen que pintar 10 casas, donde cada una ocupa 30 latas de pintura.
La gerencia liberó 20 latas por cada casa, para controlar el avance.
Inicialmente se compraron en una OC 200 latas (todo lo liberado en ese momento).
Los contratistas pidieron en total 215 latas.
La gerencia liberó las restantes 10 latas por cada casa.
¿Cuántas latas se deben comprar en este momento?
R: 115 (las 100 por avance + 15 de reposición de sobrecarga)

Se entregarán las siguientes tablas:

- Los **budget**, **activity** y **material** (id, name)
- La **cantidad cubicada/presupuestada** de cada material, por presupuesto y partida (id, material_id, budget_id, activity_id, quantity)
- La **cantidad liberada** por la gerencia de cada material, por presupuesto y partida (id, material_id, budget_id, activity_id, quantity)

- La **cantidad pedida** en tickets de cada material, por presupuesto y partida.
  Cabecera: (id, …, activity_id)
  N Detalles: (id, parent_id, budget_id, material_id, quantity)
- La **cantidad comprada** de cada material en órdenes de compra.
  (id, fecha, proveedor,..., material_id, quantity)

Recursos

Se les enviará:

- Este documento actualizado
- Los 100 casos de prueba de la parte 2
- Una BBDD Postgres 13, con las tablas y datos ya creados.

Fecha tentativa:

Recordar:

Subir las 3 partes a un repositorio público y compartir algunas horas antes de la 2da reunión