

# Clojure - Petit LP

Volem fer un petit intèrpret capaç d'avaluar expressions aritmètiques simples i executar 2 accions simples.

## Expressions:

Les expressions seran combinacions de les comandes següents amb parentesis:

Comanda	Exemple	Explicació
<code>:val</code>	<code>(:val 2)</code> 👉 2	Valor
<code>:var</code>	<code>(:var "a")</code> 👉 2	Valor d'una taula de símbols
<code>:add</code>	<code>(:add (:val 1) (:val 2))</code> 👉 3	Suma
<code>:sub</code>	<code>(:add (:val 4) (:val 2))</code> 👉 2	Resta
<code>:mul</code>	<code>(:add (:val 1) (:val 2))</code> 👉 2	Multiplicació
<code>:div</code>	<code>(:add (:val 2) (:val 1))</code> 👉 2	Divisió

## Accions:

Els programes seran llistes de les accions següents:

Comanda	Exemple	Explicació
<code>:view</code>	<code>(:view (:val 2))</code> 👉 2	Mostra un valor
<code>:ass</code>	<code>(:ass "a" (:val 2))</code>	Assignació a una taula de símbols

Un exemple de programa seria:

```
(:ass "a" (:val 2))
(:view (:div (:var "a") (:val 0))) 👉 >> Divisió per zero!
(:view (:add (:var "a") (:val 1))) 👉 >> 3
```

Taula de símbols final : {"a" 2}

## Enunciat:

- Definiu una funció `interpretar` que interpreti un programa com l'anterior i que sigui capaç de processar tot el que està especificat en les taules anteriors.

## Joc de proves:

```
(interpretar '(:ass "a" (:val 2))
              (:view (:div (:var "a") (:val 0)))
              (:view (:add (:var "a") (:val 1)))))
```



```
>> Divisió per zero!
```

```
>> 3
```

```
{"a" 2}
```

## Regles del joc:

- Heu de fer el treball en parelles.
- Heu de lliurar un arxiu `clj` amb el nom i cognoms dels dos membres del grup en un comentari a la part de dalt de l'arxiu.
- Lliureu només un dels membres del grup.