Instrucciones

Implementar un motor de expresiones regulares.

Requerimientos

- Los operadores soportados serán: concatenación, disjunción (`|`) y estrella de Kleene (*)
- Las expresiones regulares se especificaran con notación Lisp. Por ejemplo la expresión `(a|b|c)*aab(a|b|c)*` se representa con la siguiente estructura:

  ```
  (+ (* (| #\a #\b #\c)) (+ #\a #\a #\b) (* (| #\a #\b #\c)))
  ```

- Proveer una función `(compile expr)` que reciba una expresión regular y retorne su correspondiente autómata de estados finito determinístico.
- Proveer una función auxiliar que transforme una expresión regular a un autómata de estados finito no determínistico `(regex->nfa expr)`.
- Proveer una función auxiliar que transforme un autómata de estados finito no determinístico a un autómata de estados finito determinístico `(nfa->dfa nfa)`.
- Proveer una función `(accept string dfa)` que reciba una hilera y un autómata de estados finito determinístico y retorne un booleano indicando si la hilera fue reconocida o no.
- Ejemplo: `(accept "caabbc" (compile '(+ (* (| #\a #\b #\c)) (+ #\a #\a #\b) (* (| #\a #\b #\c))))` retorna `#t`.

Restricciones

- La solución se debe programar en Racket.
- El programa será evaluado con pruebas automatizadas, por tanto debe respetar a cabalidad las interfaces especificadas.
- Debe mantener su solución dentro del paradigma funcional.

Rubros

- `(regex->nfa expr) - 30 pts`
- `(nfa->dfa nfa) - 45 pts`
- `(compile expr) - 5 pts`
- `(accept string dfa) - 20 pts`
- Si una función no se evalúa correctamente, automáticamente pierde el 50% del puntaje correspondiente.