Mini Project

Idea :

* Regular expressions are patterns used to describe the lexical parts of language, such as numbers and identifiers.

* Strings matching these expressions can be detected by non-deterministic finite state automata's(NFA's) which can be transformed (more efficiently) into deterministic (DFSA) and indeed optimal forms of DFSA.

* Idea of project is to give an FSA for any regex given as input and vice versa. Additionally along with NFA, we can also provide the user with a DFSA and optimized DFSA.

Haskell Features being utilized :

* We will be utilizing pattern matching abilities of haskell to realize a regex.

* The states of FSA needs to have an ordering and equality defined in them. Nothing better than type-classes in haskell will do it!

* Modularisation: The system will be split across several modules.

* Polymorphism: The states of FSA can be represented by objects of any type.