G1:

Formula ::= Formula ‘ <->’ Formula | ImpTerm

ImpTerm ::= ImpTerm ‘ -> ‘ ImpTerm | OrTerm

OrTerm ::= OrTerm ‘\/’ OrTerm | AndTerm

AndTerm ::= AndTerm ‘/\’ AndTerm | NotTerm

NotTerm ::= ‘!’ NotTerm | Factor

Factor ::= ‘(‘ Formula ‘)’ | ‘T’ | ‘F’ | Ident

G2:

Formula ::= ImpTerm ‘<->’ Formula | ImpTerm

ImpTerm ::= OrTerm ‘->’ ImpTerm | OrTerm

OrTerm ::= AndTerm ‘\/’ OrTerm | AndTerm

AndTerm ::= NotTerm ‘/\’ AndTerm | NotTerm

NotTerm ::= ‘!’ NotTerm | Factor

Factor ::= ‘(‘ Formula ‘)’ | ‘T’ | ‘F’ | Ident

I ended up using a lot of functions and applying all of the monad, applicative, alternative stuff from the slides for tokenization, space handling and basically fetching
the Boolean statements into the Data Prop. I ran into issues with the Parser type and it was daunting to try to keep track of what I am actually returning,

The Program can parse any string, is able to check for errors when it does not pattern match an expected statement