```
-- G1
-- G1
-- Formula ::= Formula '<->' Formula | ImpTerm
-- ImpTerm ::= ImpTerm '->'  ImpTerm | AndTerm
-- AndTerm ::= AndTerm '\/'  AndTerm | OrTerm
-- OrTerm  ::= OrTerm  '/\\'  OrTerm  | Factor
-- Factor  ::= '(' Formula ')' | 'T' | 'F' | Ident

-- G2
-- Formula ::= ImpTerm '<->' Formula | ImpTerm
-- ImpTerm ::= AndTerm '->'  ImpTerm | AndTerm
-- AndTerm ::= OrTerm  '\/'  AndTerm | OrTerm
-- OrTerm  ::= Factor  '/\\'  OrTerm  | Factor
-- Factor  ::= '(' Formula ')' | 'T' | 'F' | Ident
```

The issues I had using these grammars. This is most likely due to the fact that these symbols do not exists normally in haskel. Lets say if we had math symbols such as +,-,*,^,/ etc. this would work much better. Due to this reason we made our own symbol identifier.

```
import Data.Char (isSpace, isLower, isAlphaNum)
import Control.Applicative ( Applicative(pure, (<*>)), Alternative((<|>), empty,
many) )
import System.Environment (getArgs)
import Prelude
import System.IO ()
```

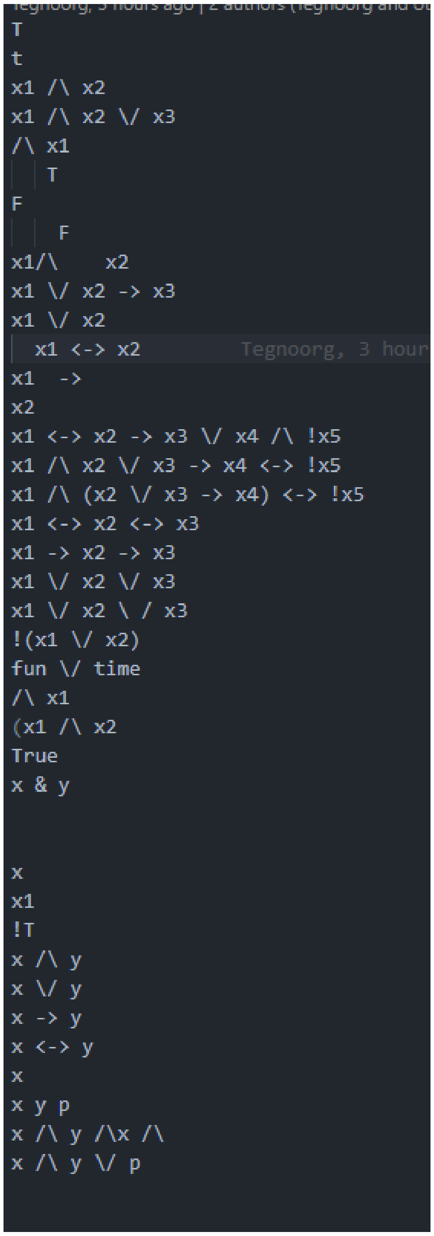
For the imports I tried to import only the things that I needed/as vscode recommended. Especially for the Const variable type. There was a collision with how Control.Applicatives read Const and Main. So due to this only calling the stuff I need from Control.Applicative's was needed.

```
parse :: Parser a -> String -> [(a, String)]
parse (P p) = p
```

I used currying to make this go from parse (P p) input = p input. This makes it shorter and nicer to look at for me.

There were other spots where I could have made the program more short and concise but it was using methods that I do not fully understand. Such as var = do Var <$> identifier. I would like to learn more about this in the future so I can use it. Overall vscode recommended ways the program could be more concise and the only reason I did not opt for those methods was due to the more concise version using the <$> operator. Due to my lack of understanding of such operator I opted out form using those methods.

Some test cases that were posted in the computer science discord from the school that I used to verify my program.



```
Tegnoorg, 3 hours ago | 2 authors (Tegnoorg and ot
T
t
x1 /\ x2
x1 /\ x2 \/ x3
/\ x1
   T
F
    F
x1/\    x2
x1 \/ x2 -> x3
x1 \/ x2
  x1 <-> x2        Tegnoorg, 3 hour
x1  ->
x2
x1 <-> x2 -> x3 \/ x4 /\ !x5
x1 /\ x2 \/ x3 -> x4 <-> !x5
x1 /\ (x2 \/ x3 -> x4) <-> !x5
x1 <-> x2 <-> x3
x1 -> x2 -> x3
x1 \/ x2 \/ x3
x1 \/ x2 \ / x3
!(x1 \/ x2)
fun \/ time
/\ x1
(x1 /\ x2
True
x & y


x
x1
!T
x /\ y
x \/ y
x -> y
x <-> y
x
x y p
x /\ y /\x /\
x /\ y \/ p
```