First, we want some imports:

```
module Main where
import Text.Megaparsec
import Text.Megaparsec.Char
import qualified Text.Megaparsec.Char.Lexer as L
import Data.Text
import Data.Void
import Control.Monad (void)
```

Then, we add some boilerplate:

```
sc :: Parser ()
sc = void $ many spaceChar

lexeme :: Parser a -> Parser a
lexeme = L.lexeme sc

symbol :: Text -> Parser Text
symbol = L.symbol sc

packSymbol :: String -> Parser Text
packSymbol = symbol . pack

parens :: Parser a -> Parser a
parens = between (packSymbol "(") (packSymbol ")")
```

The input file contains a description of a single formula in NNF using a very simplified SMT-LIB format following grammatical rules:

```
<formula> ::= '(' 'and' <formula> <formula> ')'
            | '(' 'or' <formula> <formula> ')'
            | '(' 'not' <variable> ')'
            | <variable>
```

We rewrite those into haskell:

```
formula :: Parser Formula
formula = parens $ do packSymbol "and" >> AndFormula <$> formula <*> formula
    <|> do packSymbol "or" >> OrFormula <$> formula <*> formula
    <|> do packSymbol "not" >> NotFormula <$> variable

variable :: Parser Variable
variable = lexeme $ (pure <$> letterChar) <> many alphaNumChar
```

And then we define the supporting types:

```
type Parser = Parsec Void Text

data Formula = AndFormula Formula Formula
    | OrFormula Formula Formula
    | NotFormula Variable

type Variable = String
```