Name:

CS 421 — LL Parsing

Mattox Beckman

Examples

Example 1)

$$
\begin{aligned}
E \rightarrow\ & T + E \\
\mid\ & T * E \\
\mid\ & i
\end{aligned}
$$

Example 2)

$$
\begin{aligned}
E \rightarrow\ & X + E \\
\mid\ & Y * E \\
X \rightarrow\ & a - E \\
Y \rightarrow\ & a \,/\, E
\end{aligned}
$$

Example 3)

$$
\begin{aligned}
E \rightarrow\ & E + T \\
\mid\ & E * T \\
\mid\ & i
\end{aligned}
$$

Problem 1)

$$
\begin{aligned}
T \rightarrow\ & F + T \\
\mid\ & F \\
F \rightarrow\ & F * A \\
\mid\ & A \\
A \rightarrow\ & Ni \\
N \rightarrow\ & iN \\
\mid\ & j \\
\mid\ & \epsilon
\end{aligned}
$$

1 Writing Parsers

This code is in your repository. Try to complete the code to implement the grammar.

```
import Text.Regex.TDFA

main :: IO ()
main = someFunc

isInt :: String -> Bool
isInt i = i =~ "[0-9]+"

isSymbol :: String -> String -> Bool
isSymbol s v = s == v

parseSymbol s (x:xs) =
  if s == x
     then (s,xs)
     else error $ "Parse error, expected " ++ s ++ " but got " ++ x ++ "."

-- Grammar
--
-- E -> + E E
--    | int
--    | var
--    | let var = E in E end

data Exp = PlusExp Exp Exp
         | IntExp Integer
         | VarExp String
         | LetExp String Exp Exp
    deriving Show

parse xx = parseE (words xx)

parseE ("+":xs) =
  let (e1,r1) = parseE xs
      (e2,r2) = parseE r1
   in (PlusExp e1 e2, r2)

parseE (x:xs) | isInt x =
                (IntExp (read x), xs)
```

```
Main> parse "+ 2 4"
(PlusExp (IntExp 2) (IntExp 4),[])
```