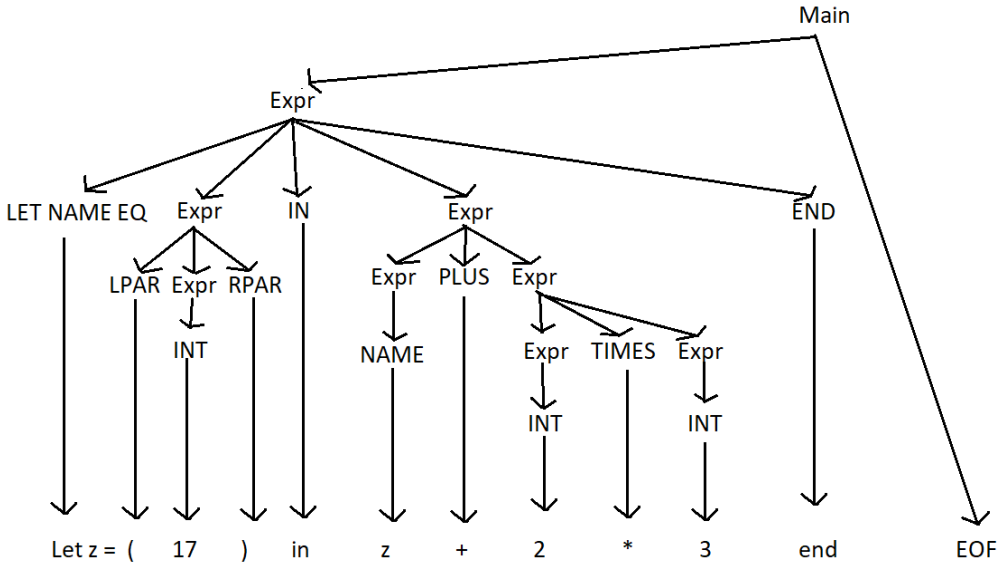
3.3

```
Main ::= Expr EOF                              rule A
Expr ::= NAME                                  rule B
       |  CSTINT                               rule C
       |  MINUS CSTINT                         rule D
       |  LPAR Expr RPAR                       rule E
       |  LET NAME EQ Expr IN Expr END         rule F
       |  Expr TIMES Expr                      rule G
       |  Expr PLUS  Expr                      rule H
       |  Expr MINUS Expr                      rule I
```

	Main
A	**Expr** EOF
F	let z = **Expr** in **Expr** end EOF
H	let z = **Expr** in **Expr** + **Expr** end EOF
G	let z = **Expr** in **Expr** + **Expr** * **Expr** end EOF
C	let z = **Expr** in **Expr** + **Expr** * 3 end EOF
C	let z = **Expr** in **Expr** + 2 * 3 end EOF
B	let z = **Expr** in z + 2 * 3 end EOF
E	let z = (**Expr**) in z + 2 * 3 end EOF
C	let z = (17) in z + 2 * 3 end EOF

3.4



3.5

Output from completing exercise 3.5

```
> open Parse;;
> fromString "1 + 2 * 3";;
val it: Absyn.expr = Prim ("+", CstI 1, Prim ("*", CstI 2, CstI 3))
```

3.6

The function compString can be found in Expr.fs line 337.
Output from our compString function:

```
val it: sinstr list =
  [SPop; SSwap; SAdd; SMul; SCstI 3; SCstI 2; SVar 0; SCstI 17]
```

3.7

See files: *ExprLex.fsl, ExprPar.fsy & Absyn.fs* for our solution.

```
> fromString "if 1 then 2 else 3";;
val it: Absyn.expr = If (CstI 1, CstI 2, CstI 3)
```