Body = SentAndCmt* + Return

SentAndCmt = Sentence | Comment

Comment = “#” CommentCont “⟨\⟩n”

Sentence = TypeDefinition | NonTypeDefSent “;”

NonTypeDefSent = VarDef | MetaExpression

VarDef = Type VarName “=” (NonLambda | VarDefLambda)

Pattern = “(“ TypeName VarName+ “)” | TypeName

NonLambda = Expression | IfClause | Matching

MetaExpression = NonLambda | Lambda

TypeDefinition = Record | Union

Record = TypeName “(” TypeName AttrName “)”

Union = UnionUnit (“|” UnionUnit)+

UnionUnit = TypeName | Record

TypeName = Iden

Lambda = “lmd” “(“ TypeVarPair (“,” TypeVarPair)* “)” LambdaBody

TypeVarPair = TypeName VarName

LambdaBody = NonTypeDefSent* Return

Return = “return” + (Expression | “none”)

IfClause = “if” Expression “{” NonTypeDefSent* “}” “else” ”{” NonTypeDefSent* “}”

Matching = “match” VarName “{” “as” Pattern “=>” NonTypeDefSent+ “}”

VarName = Iden

*Iden = [_ 0-9A-Za-z][_ 0-9A-Za-z]**

CommentCont = ⟨\⟩s+

Expression = +=*/. ** // % f(x)

Expression = Base (“**” Base)*

Base = Modulee ((“%”|“//”) Modulee)*

Modulee = Factor ((“*” | “/”) Factor)+

Factor = Term((“+”|”-”) Term)*

Term = CallItem(”(” CallItem “)”)?

CallItem = “(” Expression “)” | Iden | Number | Bool | String

Number = Int | Flo

Int = [+-]?⟨*d*⟩+

Flo = [+-]?⟨*d*⟩+.⟨*d*⟩+

Bool = True | False

String = ⟨\ ″⟩StrCont⟨\ ″⟩

StrCont = (非⟨\⟩|2個⟨\⟩)