Compiler Design tutorials

Consider the following CFG

Terminals

global	end	def	;	:	,	int	char	=
read	print	if	else	while	or	and	not	(
)	+	-	*	/	<	==	identifier	1
2	3	4	5	6	7	8	9	0

Non terminals

program	declList	decl	typeList	varList	var	type
stmtList	stmt	assignmentStmt	readStmt	printStmt	ifStmt	elsePart
whileStmt	returnStmt	bExp	exp	number		

Start Symbol

program

Production Rules

program → global declList stmtList end
decllist → decl declList | ε
decl → def typeList end
typeList → typeList ; varList : type | varList : type
varList → var , varList | var
var → identifier
type → int | char
stmtList → stmtlist ; stmt | stmt | ε
stmt → assignmentStmt | readStmt | printStmt | ifStmt | whileStmt
assignmentStmt → id = exp
readStmt → read id
printStmt → print exp
ifStmt → if bexp : stmtList elsePart end
elsePart → else stmtList | ε
whileStmt → while bexp : stmtList end
bexp → bexp or bexp | bexp and bexp | not bexp | (bexp) | exp < exp | exp == exp
exp → exp + exp | exp – exp | exp * exp | exp / exp | (exp) | identifier | number
number → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Reserved words

global	end	def	bool	else	false	if	int	true	while
print	or	and	char	read					

Identifier

Character string of length at least one.

Implement the scanner, recursive-descent parser and symbol table for the given grammar.

Test input

```
global
   def
      a:int;
      b:int;
   end
   a = 1;
   b = 7;
   if a < b :
      print a;
   else
      print b;
   end;
   while a < 5:
      b = b * 2;
   end;
   print b;
end;
```