

1	S'	S	S'.IR = S.IR print(S'.IR)
2	S	<Body>	S.IR = <Body>.IR
3	S	<Body> S	S.IR = <Body>.IR + S.IR
4	<Body>	<Decl>	.IR = <D>.IR
5	<Decl>	<VarDecl>	<D>.IR = <VD>.IR
6	<Decl>	<FuncDecl>	<D>.IR = <FD>.IR
7	<VarDecl>	int ID ;	ID.valType = 'int' IR = 'ID = 0' insert into var symbol table
8	<VarDecl>	int ID = <Exprsn> ;	ID.valType = 'int' if <E>.valType is not 'int': ERROR IR = 'ID = <E>.val' insert into var symbol table
9	<VarDecl>	float ID ;	ID.valType = 'float' IR = 'ID = 0.0' insert into var symbol table
10	<VarDecl>	float ID = <Exprsn> ;	ID.valType = 'float' if <E>.valType is not 'float': ERROR IR = 'ID = <E>.val' insert into var symbol table
11	<FuncDecl>	int ID (<FormalParams>) <StmntBlock>	<FD>.returnType != <SB>.rT: ERROR <FD>.IR = 'def func(...) + <SB>.IR + '}' <FD>.returnType = 'int' <FD>.paramName = <FP>.paramName <FD>.paramType = <FP>.paramType insert into func symbol table push <SB>.innerVarAmount vars from var symbol table
12	<FuncDecl>	float ID (<FormalParams>) <StmntBlock>	<FD>.returnType != <SB>.rT: ERROR <FD>.IR = 'def func(...) + <SB>.IR + '}' <FD>.returnType = 'float' <FD>.paramName = <FP>.paramName <FD>.paramType = <FP>.paramType insert into func symbol table push <SB>.innerVarAmount vars from var symbol table
13	<FuncDecl>	void ID (<FormalParams>) <StmntBlock>	<FD>.returnType != <SB>.rT: ERROR <FD>.IR = 'def func(...) + <SB>.IR + '}' <FD>.returnType = 'void' <FD>.paramName = <FP>.paramName <FD>.paramType = <FP>.paramType insert into func symbol table push <SB>.innerVarAmount vars from var symbol table
14	<FormalParams>	<ParamList>	<FP>.paramType = <PL>.paramType <FP>.paramName = <PL>.paramName
15	<FormalParams>	void	<FP>.paramType = [] <FP>.paramName = []
16	<FormalParams>	ε	<FP>.paramType = [] <FP>.paramName = []
17	<ParamList>	<Param>	<PL>.paramType.putHead(<P>.type) <PL>.paramName.putHead(<P>.name)
18	<ParamList>	<Param> , <ParamList>	<PL1>.paramType = <PL2>.paramType <PL1>.paramName = <PL2>.paramName <PL1>.paramType.putHead(<P>.type) <PL1>.paramName.putHead(<P>.name)
19	<Param>	int ID	<P>.type = 'int' <P>.name = ID.name
20	<Param>	float ID	<P>.type = 'float' <P>.name = ID.name
21	<StmntBlock>	{ <Stmts> }	<SB>.IR = <S>.IR <SB>.returnType = <S>.returnType <SB>.innerVarAmount = <S>.vA
22	<Stmts>	<Stmnt> <Stmts>	<Stmts1>.IR = <Stmnt>.IR + <Stmts2>.IR if <S> <Stmts2>.rT equal: <Stmts1>.rT = <Stmnt>.rT else: ERROR <Stmts1>.innerVarAmount = <Stmnt>.vA + <Stmts2>.vA
23	<Stmts>	<Stmnt>	<Stmts>.IR = <S>.IR <Stmts>.returnType = <S>.returnType <Stmts>.innerVarAmount = <Stmnt>.innerVarAmount
24	<Stmnt>	<VarDecl>	<S>.IR = <VD>.IR <S>.returnType = 'void' <Stmnt>.innerVarAmount += 1 (default is 0)
25	<Stmnt>	<IfStmnt>	<S>.IR = <IS>.IR <S>.returnType = <IS>.returnType
26	<Stmnt>	<WhileStmnt>	<S>.IR = <WS>.IR <S>.returnType = <WS>.returnType
27	<Stmnt>	<ReturnStmnt>	<S>.IR = <RS>.IR <S>.returnType = <RS>.returnType
28	<Stmnt>	<AssignStmnt>	<S>.IR = <AS>.IR <S>.returnType = 'void'
29	<AssignStmnt>	ID = <Exprsn> ;	if ID not in var table: ERROR if ID and <E>.valType not match: ERROR IR = 'ID = <E>.val'
30	<ReturnStmnt>	return <Exprsn> ;	IR = 'ret <E>.val ;' <RS>.returnType = <E>.varType
31	<ReturnStmnt>	return ;	IR = 'ret ;' <RS>.returnType = 'void'
32	<WhileStmnt>	while (<Exprsn>) <StmntBlock>	<WS>.returnType = <SB>.returnType <WS>.IR = 'L1:' + <E>.IR + 'll(<E>.val != 1) goto L2:' + <SB>.IR + 'goto L1' + 'L2:' push <SB>.innerVarAmount vars from var symbol table
33	<IfStmnt>	if (<Exprsn>) <StmntBlock> else <StmntBlock>	<SB1>.rT equals to <SB2>.rT: <IS>.rT = <SB1>.rT else: ERROR <IF>.IR = <E>.IR + 'if (<E>.val != 1) goto L1:' + <SB1>.IR + 'goto L2' + 'L1:' + <SB2>.IR + 'L2:'
34	<IfStmnt>	if (<Exprsn>) <StmntBlock>	<IS>.returnType = <SB>.returnType <IF>.IR = <E>.IR + 'if (<E>.val != 1) goto L1:' + <SB>.IR + 'L1:' push <SB>.innerVarAmount vars from var symbol table
35	<Exprsn>	<AddExprsn>	<E>.val = <A>.val <E>.valType = <A>.valType
36	<Exprsn>	<AddExprsn> < <Exprsn>	IR = 'newTemp1 = (<A>.val < <E2>.val); if newTemp1 goto L1 newTemp2 = 0 goto L2 L1: newTemp2 = 1 L2:' newTemp.valType = 'int' <E1>.val = newTemp <E1>.valType = newTemp.valType
37	<Exprsn>	<AddExprsn> <= <Exprsn>	IR = 'newTemp1 = (<A>.val <= <E2>.val); if newTemp1 goto L1 newTemp2 = 0 goto L2 L1: newTemp2 = 1 L2:' newTemp.valType = 'int' <E1>.val = newTemp <E1>.valType = newTemp.valType
38	<Exprsn>	<AddExprsn> > <Exprsn>	IR = 'newTemp1 = (<A>.val > <E2>.val); if newTemp1 goto L1 newTemp2 = 0 goto L2 L1: newTemp2 = 1 L2:' newTemp.valType = 'int' <E1>.val = newTemp <E1>.valType = newTemp.valType
39	<Exprsn>	<AddExprsn> >= <Exprsn>	IR = 'newTemp1 = (<A>.val >= <E2>.val); if newTemp1 goto L1 newTemp2 = 0 goto L2 L1: newTemp2 = 1 L2:' newTemp.valType = 'int' <E1>.val = newTemp <E1>.valType = newTemp.valType
40	<Exprsn>	<AddExprsn> == <Exprsn>	IR = 'newTemp1 = (<A>.val == <E2>.val); if newTemp1 goto L1 newTemp2 = 0 goto L2 L1: newTemp2 = 1 L2:' newTemp.valType = 'int' <E1>.val = newTemp <E1>.valType = newTemp.valType
41	<Exprsn>	<AddExprsn> != <Exprsn>	IR = 'newTemp1 = (<A>.val != <E2>.val); if newTemp1 goto L1 newTemp2 = 0 goto L2 L1: newTemp2 = 1 L2:' newTemp.valType = 'int' <E1>.val = newTemp <E1>.valType = newTemp.valType
42	<AddExprsn>	<Item> + <AddExprsn>	if <I> <A2> type not match: ERROR IR = 'newTemp = <I>.val + <A2>.val' <A1>.val = newTemp <A1>.valType = newTemp.valType
43	<AddExprsn>	<Item> - <AddExprsn>	if <I> <A2> type not match: ERROR IR = 'newTemp = <I>.val - <A2>.val' <A1>.val = newTemp <A1>.valType = newTemp.valType
44	<AddExprsn>	<Item>	<A>.val = <Item>.val <A>.valType = <Item>.valType
45	<Item>	<Factor> * <Item>	if <F> <I2> type don't match: ERROR IR = 'newTemp = <F>.val * <I2>.val' <I1>.val = newTemp <I1>.valType = newTemp.valType
46	<Item>	<Factor> / <Item>	if <F> <I2> type don't match: ERROR IR = 'newTemp = <F>.val / <I2>.val' <I1>.val = newTemp <I1>.valType = newTemp.valType
47	<Item>	<Factor>	<Item>.val = <Factor>.val <Item>.valType = <Factor>.valType
48	<Factor>	inum	<Factor>.val = inum.lexVal <Factor>.valType = 'int'
49	<Factor>	fnum	<Factor>.val = fnum.lexVal <Factor>.valType = 'float'
50	<Factor>	(<Exprsn>)	<Factor>.val = <Exprsn>.val <Factor>.valType = <Exprsn>.valType
51	<Factor>	ID	ID not in var table : ERROR else: <Factor>.val = ID.name <Factor>.valType = ID.varType
52	<Factor>	ID <FuncCall>	ID not in func table: ERROR <FC>.args does not match func table item: ERROR if type of args don't match: ERROR IR = 'newTemp = call ID(...)' newTemp.valType = ID.funcReturnType <F>.val = newTemp <F>.valType = newTemp.valType
53	<FuncCall>	(<ActualArgs>)	<F>.args = <Ac>.args <F>.argType = <Ac>.argType
54	<ActualArgs>	<ArgList>	<Ac>.args = <Arg>.args <Ac>.argType = <Arg>.argType
55	<ActualArgs>	void	<Ac>.args = [] <Ac>.argType = []
56	<ActualArgs>	ε	<Ac>.args = [] <Ac>.argType = []
57	<ArgList>	<Exprsn> , <ArgList>	<Arg1>.args = <Arg2>.args <Arg1>.argType = <Arg2>.argType <Arg1>.args.putHead(<E>.val) <Arg1>.argType.putHead(<E>.valType)
58	<ArgList>	<Exprsn>	<Arg>.args.putHead(<E>.val) <Arg>.argType.putHead(<E>.valType)