PROJECT #1

ID CROSS REF

WHAT TO DO:

CREATE A LISTING OF EACH ID DECL w/ ALL ITS USES

E.G.,

```
{ INT a;
  BOOL b;
  IF (b)
     a = a+1;
}
```

PRODUCES

```
1:  a(INT):  4(2)
2:  b(BOOL): 3
```

ID CROSS REF IS USED FOR REFACTORING AND DEBUGGING

WHAT YOU LEARN:

- REFRESH YOUR JAVA CODING
- LEARN COMPILER CONCEPTS LIKE
  - SCANNER
  - PARSER
  - ABSTRACT SYNTAX TREE (AST)
  - BLOCK STRUCTURED SYMBOL TABLE

WHAT IS CSX-LITE?

SMALL SUBSET OF CSX

STRUCTURE:

PROGRAM IS:

{ DECLS STMTS }

A DECL IS:

INT id; OR
BOOL id;

An id IS:

LETTER FOLLOWED BY
LETTERS & DIGITS

E.G. a xyz GoTeam CVN76

CASE IS IGNORED!

STATEMENTS ARE

(1) ASSIGNMENT:

id = EXPR;

(2) CONDITIONAL:

IF (EXPR) STMT;

(3) BLOCK:

{ DECLS STMTS }

EXPRS ARE:

(1) id

(2) INT LITERAL

1 0 777

(3) BINARY EXPR:

EXPR OP EXPR

a+b a-(b-c)

(4) OPS ARE

+ - == !=

PARENS ARE ALLOWED

(a+b) (a+(b+c))

COMMENTS ARE SINGLE LINE:

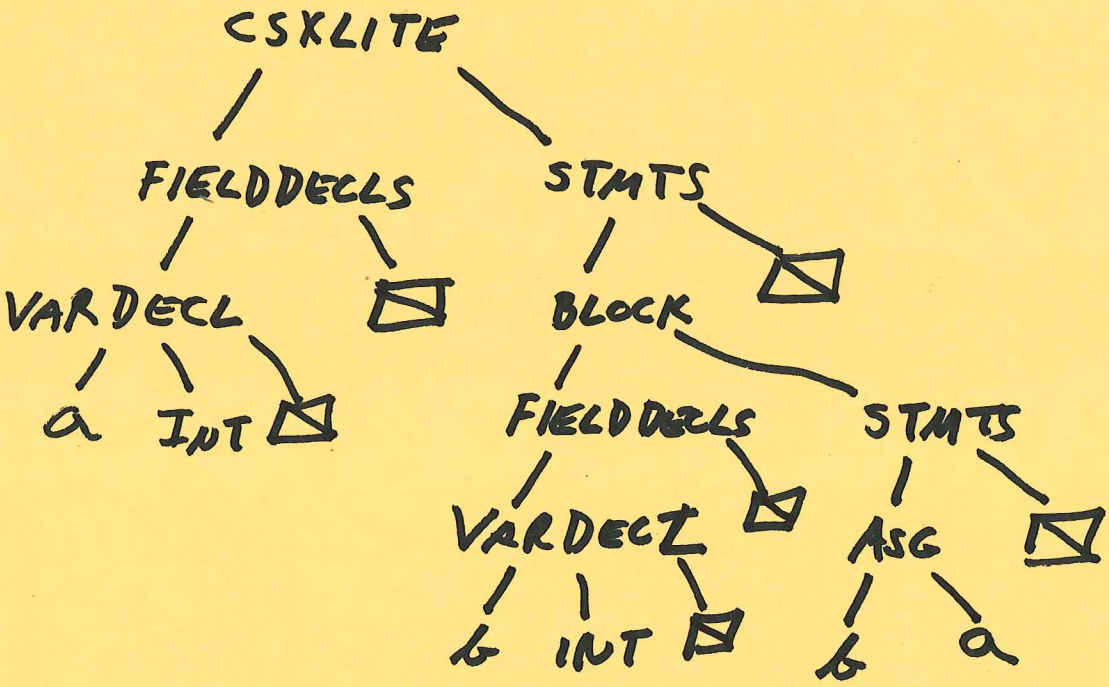
// TEXT OF COMMENT

WE GIVE YOU A WORKING CSX-LITE SCANNER THAT BUILDS A VALID AST.

YOU NEED NOT HANDLE INVALID PROGRAMS!

YOU WALK THE AST TO GATHER DECL & USE INFO

```
{ INT a;
    { INT b;
      b = a;}
}
```


CSXLITE
FIELDDECLS
STMTS
VARDECL
BLOCK
a
INT
FIELDDECLS
STMTS
VARDECL
ASG
b
INT
b
a

BLOCK STRUCTURED SYMBOL TABLES

SCOPES NEST:

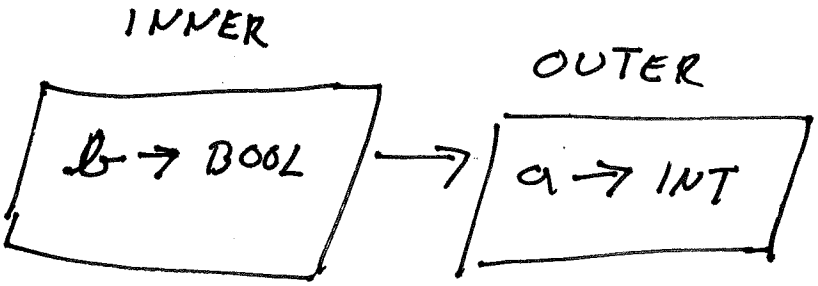
```
{ INT a
    { BOOL b
    }
    IF(b) a=1; //HERE
}
```

LOOKUP IS INNERMOST TO OUTERMOST

AT POINT "HERE", TRY INNER SCOPE, THEN OUTER ONE

EACH SCOPE HAS A SYMBOL TABLE. IT MAPS ID NAME TO ID INFO.

SYMBOL TABLES ARE LINKED TO REFLECT NESTING



TO GET STARTED WE GIVE YOU A WORKING SOLUTION TO A SIMPLER PROBLEM:

COUNT DECLS AND USES PER SCOPE (W/O SCOPING RULES!)

```
{ INT a; INT b;
  ...
    { INT c;
      a = b+c;}}
```

SCOPE 1 (LINE 1): 2 DECLS, 0 USES
SCOPE 2 (LINE 3): 1 DECL, 3 USES

↑ IGNORES SCOPING!