Checkin 15

```
 1. int a;
 2. bool f;
 3. int m(int arg){
 4.     int b;
 5.     return arg + 1;
 6. }
 7.
 8. int g(){
 9.     int c;
10.     int d;
11.     if (a){
12.         int d;
13.         int f;
14.         int g;
15.     }
16. }
```

Administrivia

- Thanks for the HOPE Award nomination!

Flipped Wednesday

Written Work #4

Topics:

- Parser generation

Written Work #4: Question 1

Draw the SLR parser table corresponding to the following grammar

```
X ::= a X b
X ::= X c Y
Y ::= d
Y ::= k X
```

Written Work #4: Question 2

Is the prior grammar parseable by an SLR parser? What evidence do you have of that?

Written Work #4: Question 3

Draw the AST for the following Levi program, using the recommended AST node types suggested in the P3 inheritance diagram.

```
a:int;
func(b:bool) int{
    if (b){
        return 1;
    }
    return 3;
}
main : () int{
    a = 7;
    return a * func(false);
}
```

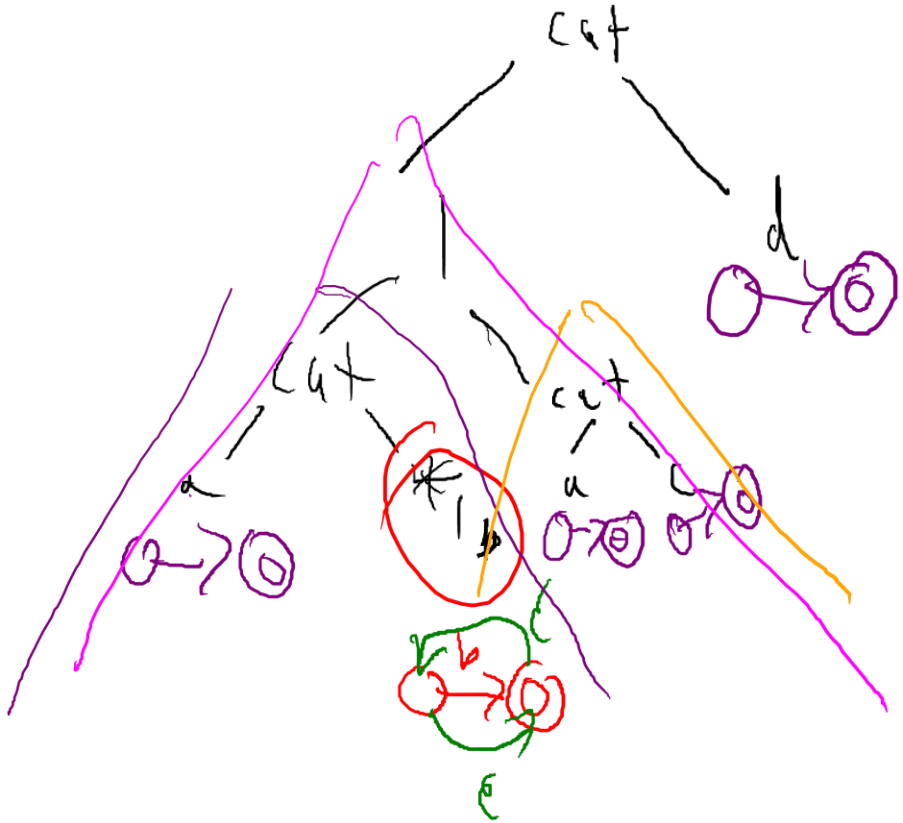
Written Work #4: Question 4

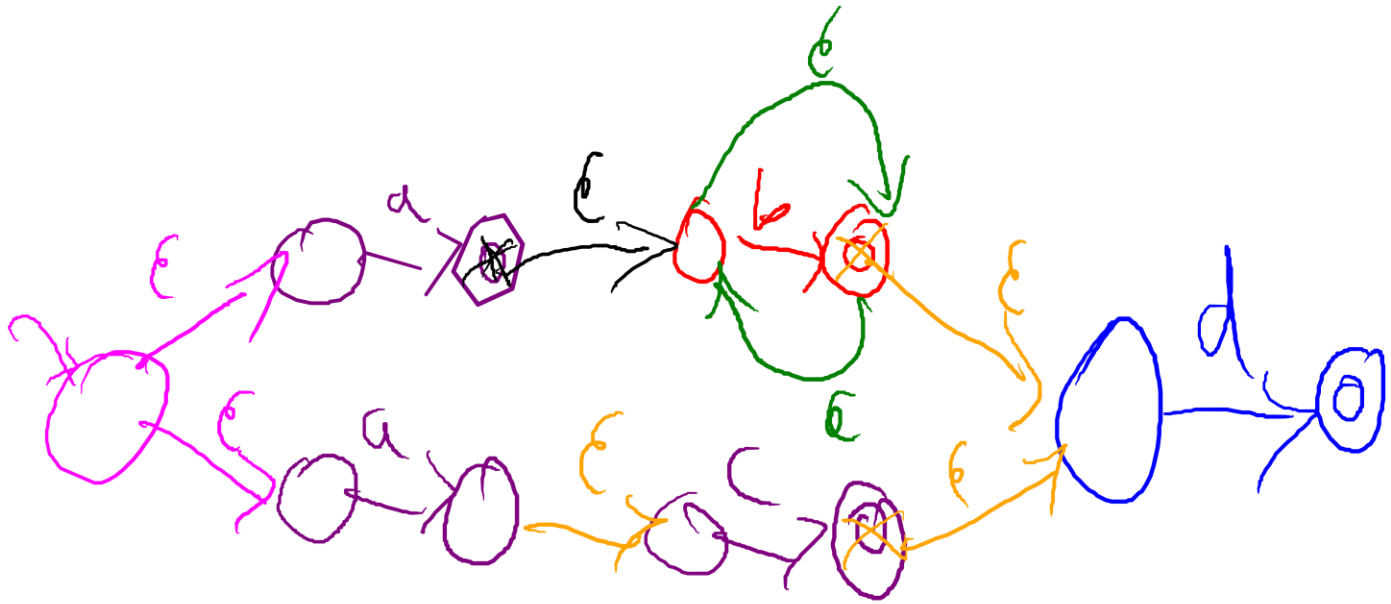
List the FIRST and FOLLOW sets for the following grammar (do not transform the grammar):

```
L ::= v m L
L ::= ε
L ::= C
L ::= D
C ::= ε
C ::= C k
D ::= C
C ::= m L
```

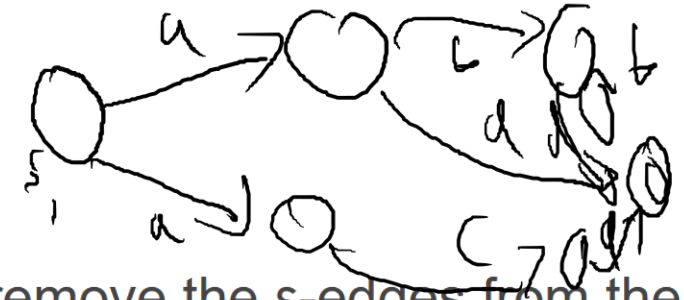
Question 2

Convert the regular expression from above into an ε-NFA (i.e. an NFA with ε-edges) using Thompson's algorithm.





Question 3

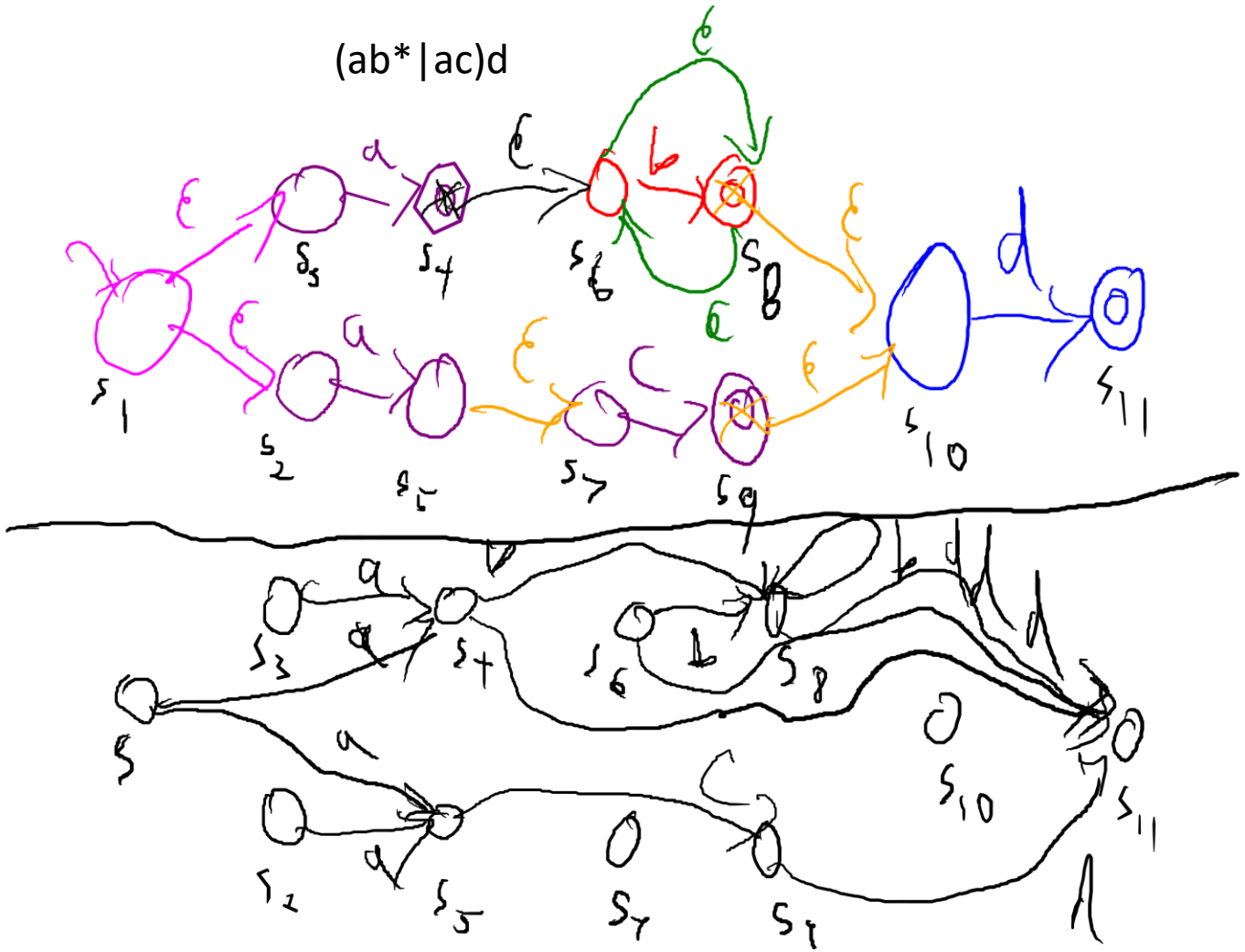


$s_{10} = \{ s_{10} \}$

$s_{11} = \{ s_{11} \}$

Use the ε-elimination technique to remove the ε-edges from the previous ε-NFA.

(ab*|ac)d



$\varepsilon\text{-}clos(s_1) = \{ s_1, s_2, s_3 \}$

$s_2 = \{ s_2 \}$

$s_3 = \{ s_3 \}$

$s_4 = \{ s_4, s_6, s_8, s_{10} \}$

$s_5 = \{ s_5, s_7 \}$

$s_6 = \{ s_6, s_8, s_{10} \}$

$s_7 = \{ s_7 \}$

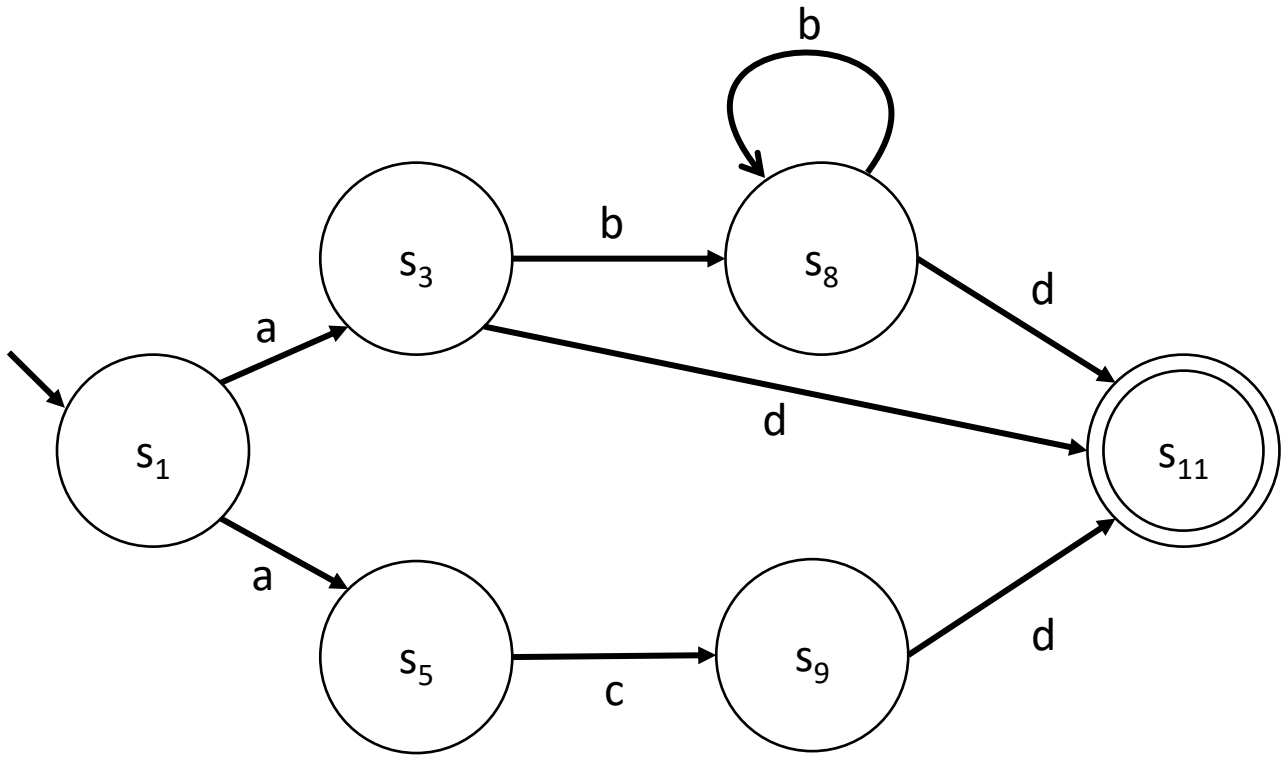
$s_8 = \{ s_8, s_6, s_{10} \}$

$s_9 = \{ s_9, s_{10} \}$

Question 3

Use the ε-elimination technique to remove the ε-edges from the previous ε-NFA.

(ab*|ac)d



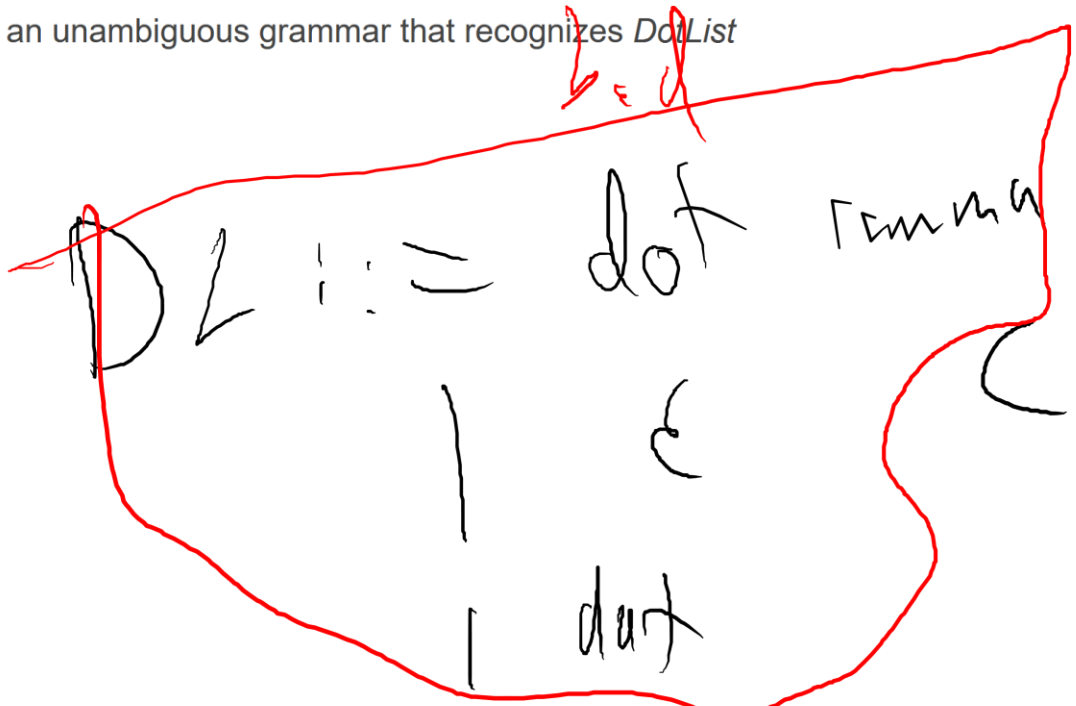
Question 4

Let *DotList* be a language such that:

- The empty string is in the language
- The single terminal **dot** is in the language
- Sequences of more than 1 **dot** terminal separated by the **comma** terminal are in the language. e.g.:
  - **dot comma dot**
  - **dot comma dot comma dot**

No other strings are in the language

Write an unambiguous grammar that recognizes *DotList*



DL ::= ε

DL ::= dot comma DL | CL

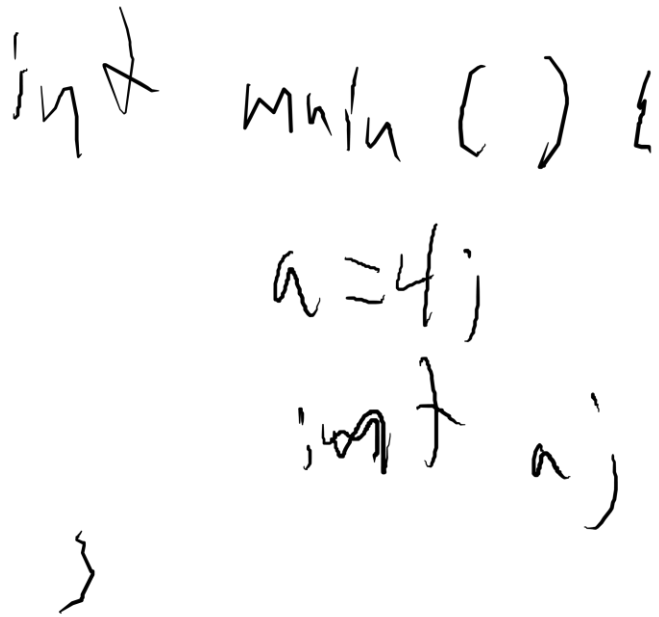
CL ::= dot comma CL | dot

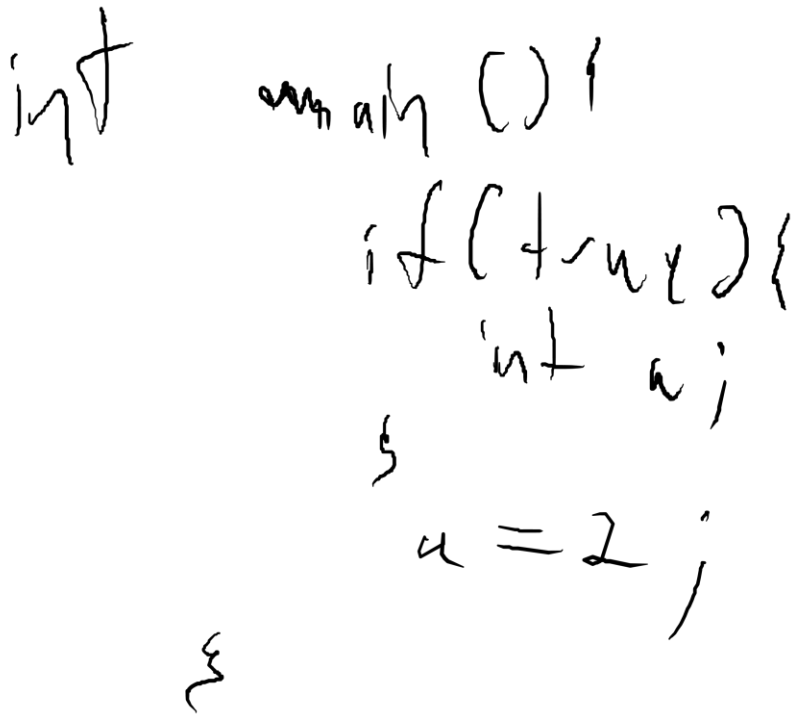
Written Work #1: Question 2

What is the purpose of the syntactic analysis component of a compiler? Give an example of an input that GCC would flag for a syntactic error.

Written Work #1: Question 3

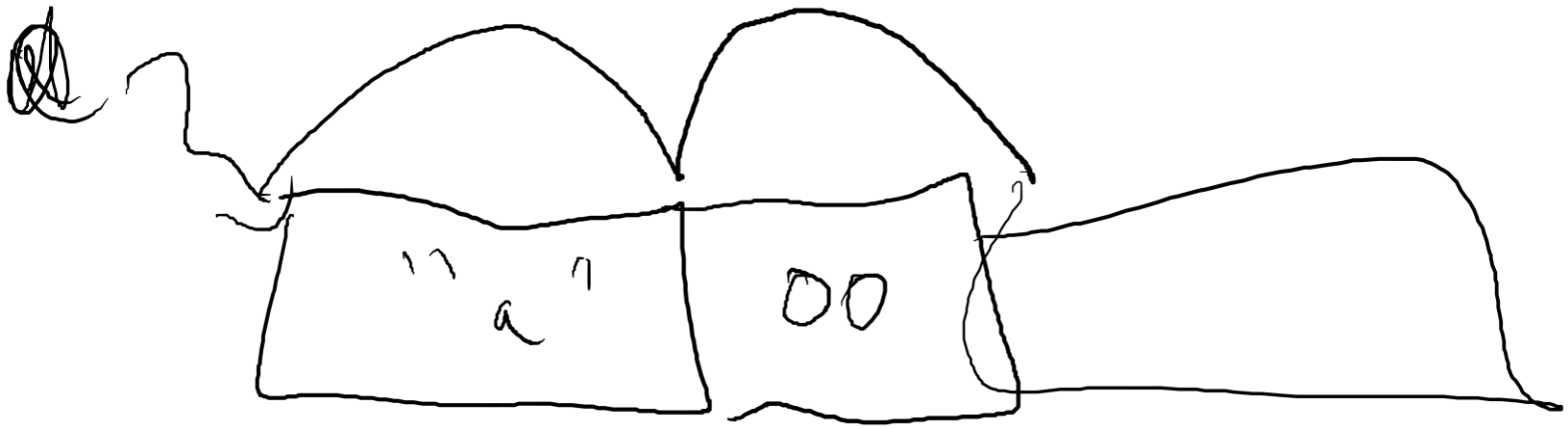
What is the purpose of name analysis in a compiler? Give an example of an input that GCC would flag for failing name analysis.

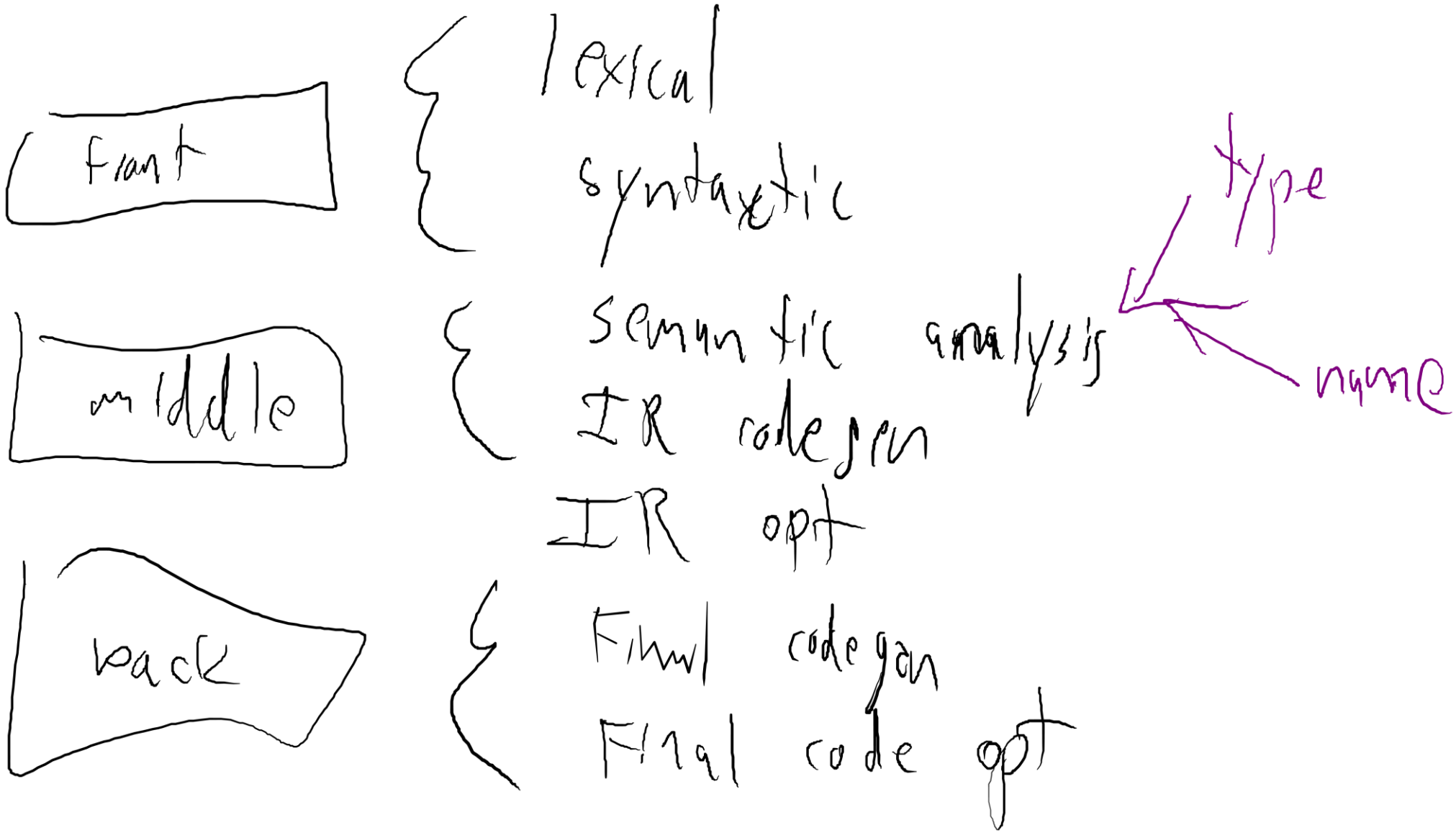




Written Work #1: Question 4

What is the purpose of type analysis in a compiler? Give an example of an input that GCC would flag for failing type analysis.




front
lexical
syntactic
middle
semantic analysis
IR codegen
IR opt
type
name
back
Final codegen
Final code opt

Create the full PEMDAS grammar