Homework 6

Due by **11PM on March 16**

Homework assignments must be done individually. Collaboration on homework assignments is *not* allowed.

Question 1:

Suppose that your grades are kept in a file with the following format:

- The file contains one or more student records.
- Each student record has one student's name, then their ID number, then zero or more grades, separated by commas.
- Each grade is an integer value followed by zero or more stars (to represent the number of days late).

The TAs decided to write a parser to read the grades in order to make it easy to summarize them. Here's a CFG for the grade file format that they came up with. The grade file is updated after every assignment is graded.

```
file → record tail
tail → file | ε
record  → NAME IDNUM optGrades
optGrades → grades | ε
grades → oneGrade | oneGrade COMMA grades
oneGrade → INTLIT optLate
optLate → stars | ε
stars → STAR | stars STAR
```

a. Although the grade file CFG given above is not LL(1), some correct inputs can be parsed by a predictive parser. This is because those inputs never cause the parser to look at a table entry that contains two or more CFG rules. However, the TAs ran into problems parsing this file at some point in the course because the CFG is not LL(1).

   i. Provide an example for the **shortest** input which the parser can successfully read (as a sequence of tokens, ending with `EOF`).
   ii. Draw the parse tree that the parser would build for the input you gave for Part (i). (Do not include the `EOF` token in the parse tree.)

b. Provide an example for the **shortest** input on which the parser fails. (i.e., a sequence of tokens that is a prefix of a valid input, but for which the parser would not know how to continue to build the parse tree top-down because it looks at a table entry that contains two or more CFG rules)? To answer this question, give all of the following:

   i. The sequence of tokens (a **prefix** of a valid input).
   ii. The (partial) parse tree that the predictive parser would have built before being stumped.
   iii. The CFG rules that the predictive parser can't choose between to continue to grow the parse tree.

c. One problem with the grade file CFG is that it has not been left factored. Find the CFG rules with a common prefix and transform them by doing left factoring.

d. Another problem with the grade file CFG is that is has immediate left recursion. Find the CFG rules that cause this and transform them to remove the left recursion.

Question 2:

This question concerns the following grammar.

```
program -> { stmts } EOF
stmts ->  stmt  stmts
       | epsilon
stmt -> ID = exp ;
      | IF ( exp ) stmt
exp -> ID tail
tail -> + exp
      | - exp
      | epsilon
```

Compute the FIRST and FOLLOW sets for the non-terminals of this grammar.