All students should read §6.3 of the textbook in preparation for lecture and the questions **in this assignment**.

Algorithms for the following questions have been presented in lecture and can be found in the book (this reading assignment). Slides and pseudo code from lecture are posted on the schedule page alongside this group assignment.

> Questions 1–3 have to do with generating grammar's CFSM, questions 4–5 focus on how a CFSM is turned into an SLR(1) table. Students should try to work one of each.
> CFSMs (item set graphs) can be pretty big, so this LGA has links to their visual PDF instead of embedding them into the document. The links are blue text and should work for any reasonable PDF viewer, other than `pdftotext(1) :)`
> Finally, for those questions where you build SLR(1) tables, I recommend putting the columns in **lexical order** by ASCII code. This will make it much easier to compare your results with your peers (which will already be difficult enough, since the only item set label your CFSMs might share is "Item Set 0").

1. (a) Calculate the full CFSM (item set graph) for the following grammar.

$$
\begin{array}{rcl}
START & \rightarrow & E\ \$ \\
E & \rightarrow & plus\ E\ E \\
 & | & num
\end{array}
$$

   (b) There are three items sets (and their incident, emanating edges) missing for the grammar below and its `CFSM linked to here` . Determine these missing nodes and edges and describe how to make the CFSM correct. **Clarification:** this is a big(ish) language, I **don't** expect you to work out all the item sets by hand. Instead, consider what structural characteristics CFSMs and their item sets have, and inspect the provided graph for them. Report back to your learning group your corrections (additions) to the CFSM to make it correct and **most importantly** how did you deduce your additions were missing?

$$
\begin{array}{rcl}
S & \rightarrow & T\ U\ C\ \$ \\
T & \rightarrow & x\ T\ y\ T \\
 & | & t \\
 & | & \lambda \\
U & \rightarrow & A\ B\ C \\
A & \rightarrow & x\ B \\
B & \rightarrow & y\ C \\
C & \rightarrow & U\ m\ n \\
 & | & \lambda
\end{array}
$$

2. Consider the following grammar and one of its item sets. Find all the item sets and connecting edges **that this item set** will generate. This turns out to be a pretty big CFSM (item set graph) — you aren't expected to calculate the whole thing! Just find the **first generation** of item sets that will have edges emanating from this one.

$$
\begin{array}{rcl}
S & \rightarrow & SUM\ \$ \\
SUM & \rightarrow & PRODUCT \\
 & | & SUM\ ADD\ PRODUCT \\
PRODUCT & \rightarrow & POWER \\
 & | & PRODUCT\ MULT\ POWER \\
POWER & \rightarrow & VALUE \\
 & | & VALUE\ EXP\ POWER \\
VALUE & \rightarrow & num \\
 & | & var \\
 & | & (\ SUM\ ) \\
ADD & \rightarrow & + \\
 & | & - \\
MULT & \rightarrow & * \\
 & | & / \\
 & | & \% \\
EXP & \rightarrow & **
\end{array}
$$

CHARLES

$PRODUCT \rightarrow PRODUCT\ MULT \bullet POWER$
$POWER \rightarrow \bullet\ VALUE$
$POWER \rightarrow \bullet\ VALUE\ EXP\ POWER$
$VALUE \rightarrow \bullet\ num$
$VALUE \rightarrow \bullet\ var$
$VALUE \rightarrow \bullet\ (\ SUM\ )$

3. Consider the grammar for question 2 on the preceding page that represents a conventional arithmetic notation.
   (a) Find "Item Set 0", the one at that would be considered the starting state of the CFSM (item set graph).
   (b) Find all the **first generation** item sets generated by your Item Set 0. Which is to say all the item sets with edges from "Item Set 0".
   (c) How many **second generation** item sets are there in the CFSM for this grammar (you don't need to find them all, just deduce how many there will be.

4. Given the grammar below, its follow sets and its correct and complete CFSM (item set graph) linked to here ,

#	Rules
1	$S \rightarrow SUM\,\$$
2	$SUM \rightarrow SUM + V$
3	$SUM \rightarrow V$
4	$V \rightarrow num$
5	$V \rightarrow (\,SUM\,)$

```
Follow(SUM,k=1) $  )  +
Follow(V,k=1) $  )  +
```

   (a) Write out the SLR(1) parsing table for the grammar.
   (b) Is this grammar "SLR(1)" or are there conflicts in the table?

5. Given the grammar below, its follow sets and its correct and complete CFSM (item set graph) linked to here ,

#	Rules
1	$START \rightarrow S\,\$$
2	$S \rightarrow A\,B$
3	$S \rightarrow a\,c$
4	$S \rightarrow x\,A\,c$
5	$A \rightarrow a$
6	$B \rightarrow b$
7	$B \rightarrow \lambda$

```
Follow(S,k=1) $
Follow(A,k=1) $  b  c
Follow(B,k=1) $
```

	a	*b*	*c*	*x*	$	*S*	*A*	*B*
0	sh-1			sh-2		sh-3	sh-4	
1		r-5	sh-5		r-5			
2	sh-6						sh-7	
3					⋆			
4		sh-9			r-7			r-2
5					r-3			
6		r-5	r-5		r-5			
7			sh-11	sh-7				
8	Reduce 1							
9					r-6			
10					r-2			

   (a) Find **all the errors** in the **faulty** SLR(1) table shown above, and be able to describe to your learning group **how you found them — no, the answer isn't "I build my own SLR(1) table and compared the two..."**
   (b) Is the grammar "SLR(1)" or are there conflicts in the table?

6. Suppose you are given a correct, un-conflicted SLR(1) table and told the grammar it is for contains only one non-terminal. Can you deduce the grammar rules from this information? How? Or why not?
   Hint: properly written grammars have a starting goal with $ at the end of all its production rules, and therefore starting goals cannot have recursive rules.
   Hint: an SLR(1) table for you two contemplate while you reason out your answer is below.

	a	*c*	*q*	*x*	$
0		sh-1	sh-2	sh-3	
1		sh-4		sh-5	
2			sh-6		
3	sh-7	sh-8			
4	sh-9				
5				sh-10	
6			sh-11		
7		sh-12			
8	sh-13				
9					sh-14
10					sh-15
11					sh-16
12					sh-17
13					sh-18
14	Reduce 3				
15	Reduce 4				
16	Reduce 5				
17	Reduce 1				
18	Reduce 2				