CS 536

Practice Midterm

Fall 2018

Answer Key

Question 1

(a) If n is the number of characters in the alphabet, then no string can be longer than n unless characters ~~unless~~ repeat. The set of all strings no longer than n is finite, and hence regular

(b) This is very similar to the CSX multi-line comment:

$<<\ (\ (>\mid\lambda)\ \mathrm{Not}(>)\ )^{*}\ >>$

Question 2

Two ways to show S is <u>not</u> regular

(i) In an old exam question we established that if S_1 and S_2 are regular, so is $S_1 - S_2$ (since $S_1 - S_2 = S_1 \cap \overline{S_2}$)

But $([^+]^+) - S$ is balanced brackets, <u>not</u> regular

(ii) Same idea as balanced brackets:

$[]$, $[[]]$, … $[^i]^i$ must all now be rejected.

Read $[$, $[[$, $[[[$, etc until two distinct prefixes, $[^i$ and $[^j$ both reach the same state

$[^i]^i$ must reach a non-accepting state

But $[^j]^i$ will reach the same state and it should be accepting (since $j \neq i$).

A contradiction!

Question 3

(a) There are many possible answers. Here is one:

```
\"\\n\"
```

(b)

```
\\(\\\\)*
```

(c) We'll do the comment in three segments.

```
Oneline = "{" [^}\n]* "}"
Twolines = "{" [^}\n]* \n  [^}\n]* "}"
Threelines =
 "{" [^}\n]* \n  [^}\n]* \n  [^}\n]* "}"

Answer = {Oneline} | {Twolines} |
{Threelines}
```

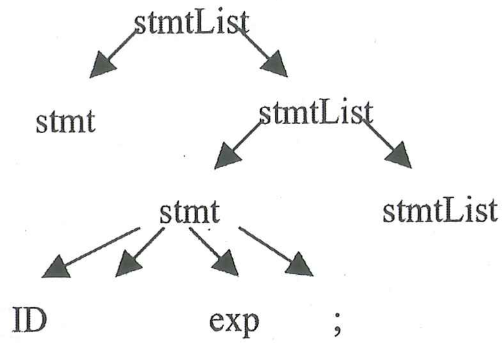
4. Below is a context-free grammar for a language of assignments that includes arrays:

1.	stmtList	→	stmt stmtList
2.		|	λ
3.	stmt	→	ID = exp ;
4.	array	→	[rowList]
5.	rowList	→	nonEmpty
6.		|	λ
7.	nonEmpty	→	row moreRows
8.	moreRows	→	; nonEmpty
9.		|	λ
10.	row	→	exp more
11.	more	→	, row
12.		|	λ
13.	exp	→	term tail
14.	tail	→	+ term tail
15.		|	λ
16.	term	→	ID
17.		|	INTLIT
18.		|	array

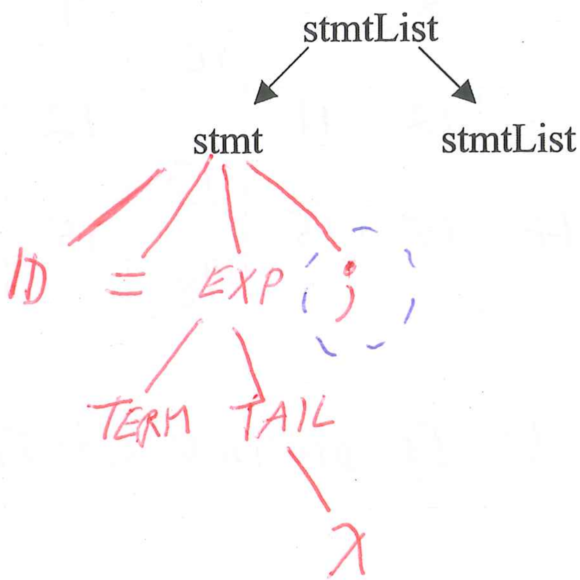
Here are the *FIRST* and *FOLLOW* sets for all of the non-terminals:

Non-terminal *X*	***FIRST(X)***	***FOLLOW(X)***
stmtList	ID	EOF
stmt	ID	ID EOF
array	[	+ , ;]
rowList	ID INTLIT [	]
nonEmpty	ID INTLIT [	]
moreRows	;	]
row	ID INTLIT [	;]
more	,	;]
exp	ID INTLIT [	, ;]
tail	+	, ;]
term	ID INTLIT [	+ , ;]

(a) Recall that terminal *t* is in *FOLLOW*(*X*) if in some partial parse tree with the start non-terminal at the root, *X* is one leaf of the tree and *t* is the next non-lambda leaf immediately to the right. For example, the following partial parse tree justifies the fact that for the CFG given above, terminal ID is in *FOLLOW*(stmt):



Complete the partial parse tree below to justify the fact that terminal ; is in *FOLLOW*(term).



(b) Fill in the parse table below using the numbers of the grammar rules rather than the rules themselves. Is the grammar LL(1)?

	ID	INTLIT	=	+	;	,	[	]	EOF
stmtList	1								2
stmt	3								
array							4		
rowList	5	5					5	6	
nonEmpty	7	7					7		
moreRows					8			9	
row	10	10					10		
more					12	11		12	
exp	13	13					13		
tail				14	15	15		15	
term	16	17					18		

<u>IS</u> LL(1) NO PREDICTION CONFLICTS

5. Consider the following grammar

$$
\begin{array}{rcl}
\mathit{File} & \rightarrow & \mathit{Record} \\
 & | & \mathit{Record\ File} \\
\mathit{Record} & \rightarrow & \textbf{name idnum } \mathit{OptGrades} \\
\mathit{OptGrades} & \rightarrow & \mathit{Grades} \\
 & | & \lambda \\
\mathit{Grades} & \rightarrow & \mathit{OneGrade} \\
 & | & \mathit{OneGrade}\ \textbf{comma}\ \mathit{Grades} \\
\mathit{OneGrade} & \rightarrow & \textbf{intlit}\ \mathit{OptLate} \\
\mathit{OptLate} & \rightarrow & \mathit{Stars} \\
 & | & \lambda \\
\mathit{Stars} & \rightarrow & \textbf{star} \\
 & | & \mathit{Stars}\ \textbf{star}
\end{array}
$$

where *File* is the start non-terminal, and symbols in **bold** are terminals.

(a) Apply the transformations learned in class to *left factor* the grammar above and write the results below. Give the entire grammar, not the just the transformed rules.

FILE → RECORD FILE1
FILE1 → FILE | λ
RECORD → NAME IDNUM OPTGRADES
OPTGRADES → GRADES | λ
GRADES → ONEGRADE G1
G1 → COMMA GRADES | λ
ONEGRADE → INTLIT OPTLATE

OPTLATE → STARS | λ
STARS → STAR
| STARS STAR

(b) If the grammar you wrote above has any immediate left recursion, apply the transformation learned in class to remove it and write the result below. You do not need to give the entire grammar; you can just give the transformed rules.

STARS → STAR S'
S' → STAR S' | λ