CSci 365: Organizations of Programming Languages

Name: ______________________

Assignment 3: 80 + 30 (optional) points

– Read the submission instructions.

Q1. [10] In the recursive-descent parsing, a direct/indirect left recursive rule disallows top-down parsing. In the following grammar, convert it to the non-left recursive rules.

$S \rightarrow Aa \mid b, \quad A \rightarrow Ac \mid Sd \mid \varepsilon$

Q2. [10] The following grammar that removed left recursive rules allows a top-down parsing. Draw a *sequence of parse tree* step by step for a sentence '*a + a * a*' using a recursive-descent top-down parsing.

$E \rightarrow T\,E', \quad E' \rightarrow +\,T\,E' \mid \varepsilon, \quad T \rightarrow F\,T', \quad T' \rightarrow *\,F\,T' \mid \varepsilon, \quad F \rightarrow (E) \mid a.$

Q3. [15] The lack of pairwise disjointness disallows top-down parsing.
(A) Get the FIRST set of each rule, then (B) Test if the given grammar rules are pairwise disjoint or not.

1) $A \rightarrow aaA \mid b \mid caB$
2) $A \rightarrow aB \mid bA \mid aBb$

Q4. [20] Given the grammar and a right sentential form 'aAcccb**b**c',

$S \rightarrow AbB \mid bAc, \quad A \rightarrow Ab \mid aBB, \quad B \rightarrow Ac \mid cBb \mid c$

1) [5] Draw a parse tree.

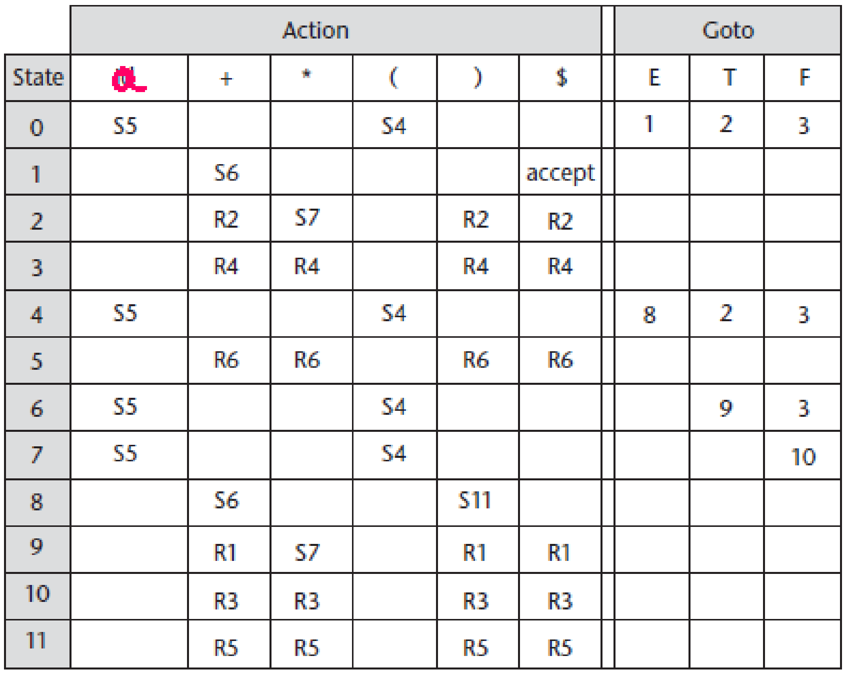
2) [15] Show the (A) phrases, (B) simple phases, and (C) the handle.

Q5. [25] The bottom-up parsing uses an LR-parser. Show a complete LR-parse, including parse stack contents, input string, and action for the string *a * (a + a)* that ends with a marker '$', using the following grammar of the expression and the parse table below. Complete your parsing in the given empty table. – Refer to the handout, LR-parsing-Action-GOTO.pdf. If you'd like to, you can write a program to implement it in Q6. The output must print the LR-parsing table with the sentential forms.

Grammar:

1. $E \rightarrow E + T$,
2. $E \rightarrow T$
3. $T \rightarrow T * F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow a$

Parse Table:



	Action						Goto		
State	a	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		S6				accept			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

LR-Parsing Table:

Stack	**Input**	**Lookahead symbol**		**Action**	**Sentential Forms**
0	*a* * (*a* + *a*) $	*a*	LR(0, *a*) = S5	Shift 5	*a* * (*a* + *a*) $
…..					
Add more rows if necessary					
…..					
0E1	$	$	LR(1, $) = accept!!		E $

Q6. [30, optional] For Q5, write a Python program to implement LR-parsing with the given input and the

Parse table. Your output must print the LR-parsing table of (stack content, input, Action). For ‘Action’, it must print ‘Shift-*n*’ or ‘Reduce-*n* – use GOTO[*m*, *s*]’ where *n* is the rule number of the given grammar, *m* is a state number, and *s* is a non-terminal symbol.

Thus, your output must print the sequence of sentential forms as well as other contents in the table.

For example, your output looks like the below:

```
stack-content  input          Lookahead    Action        Sentential Forms
                              Symbol
-------------------------------------------------------------------------------------------------
0              a * (a + a) $  a            Shift 5       a * (a + a) $

-------------------------------------------------------------------------------------------------
0a5            * (a + a) $    *            Reduce 6      F * (a + a) $

                                           GOTO(0, F)
-------------------------------------------------------------------------------------------------


...            ...            ...          ...           ...

...            ...            ...          ...           ...
-------------------------------------------------------------------------------------------------
0E1            $              $            Accept        E $
```