BRAC UNIVERSITY

Department of Computer Science and Engineering

Examination: Midterm Examination Semester: Summer 2025

Duration: 1 Hour 20 Minutes **Set - B** Full Marks:30

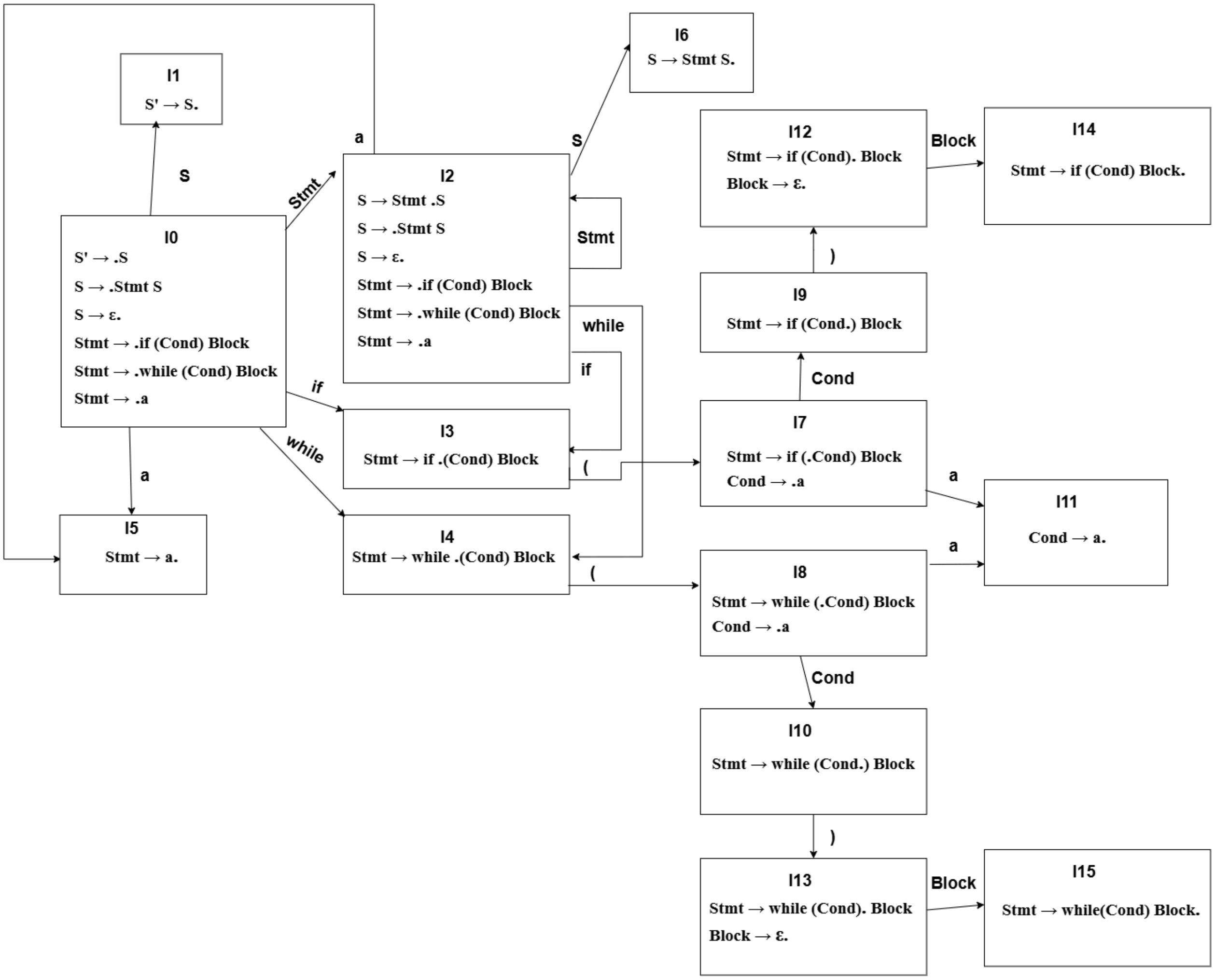
CSE 420: Compiler Design

Figures in the right margin indicate marks.

Answer all the questions

COs	Questions	Marks
CO3	1. Explain using example what configuration in an LR(0) automaton and Follows of non-terminals lead to reduce-reduce and shift-reduce conflicts. Further explain, why shift-shift conflict cannot happen?	5
CO1	2. Explain how symbol tables are used to manage scope and detect redeclaration or undefined variable errors during semantic analysis.	3
CO3	3. Consider the following grammar and along with its corresponding automaton and the SLR(1) parsing table provided below: 1. S → Stmt S 2. S → ε 3. Stmt → if (Cond) Block 4. Stmt → while (Cond) Block 5. Stmt → a 6. Block → ε 7. Cond → a *(Note that ε means empty string)*	12

Automaton:



First And Follow Set:

First(S) = {if,while,a, ε}
First(Stmt) = {if,while,a}
First(Block) = {ε}
First(Cond) = {a}

Follow(S) = {$}
Follow(Stmt) = {if, while, a, $}
Follow(Block) = {if, while, a, $}
Follow(Cond) = {) }

SLR(1) Parse Table:

State	Action						Go To			
	if	while	(	)	a	$	S	Stmt	Block	Cond
0		S4			S5		1	2		
1										
2		S4			S5					
3			S7							
4			S8							
5										
6						R1				
7					S11					9
8					S11					10
9				S12						
10				S13						
11				R7						
12		R6								
13		R6							15	
14	R3	R3			R3	R3				
15	R4	R4			R4	R4				

Using the provided context-free grammar (CFG), LR(0) automaton, and the FIRST and FOLLOW sets, first verify whether the given SLR(1) parsing table is complete. After completing the table (if necessary), demonstrate whether the input string is grammatically correct or not.

Input string: **while (a) if (a) a**

Note: You have to draw the table on your answer script.

CO4	4. Draw the parse tree of nodes – for the input sentence **(id / id + id * id)** using the following attribute grammar. Note that it is important that the ordering of nodes matches the SDD in your drawing.
	$E \rightarrow E_1 + T$ {E.node = new Node(); childList = new List();
	childList.add(T.node); childList.add(new Leaf('+'));
	childList.add(E_1.node); E.node.children = childList;}
	$E \rightarrow T$ {E.node = T.node;}
	$T \rightarrow T_1 * F$ {T.node = new Node(); childList = new List();
	childList.add(F.node); childList.add(new Leaf('*'));
	childList.add(T_1.node); T.node.children = childList;}
	$T \rightarrow T_1 / F$ {T.node = new Node(); childList = new List();
	childList.add(T_1.node); childList.add(new Leaf('/'));
	childList.add(F.node); T.node.children = childList;}
	$T \rightarrow F$ {T.node = new Node(); childList = new List();
	childList.add(F.node); T.node.children = childList;}
	$F \rightarrow id$ {F.node = new Leaf(id.lexeme);}
	$F \rightarrow (E)$ {F.node = E.node;}
		10