BRAC UNIVERSITY

Department of Computer Science and Engineering

Examination: Midterm Examination Semester: Summer 2025
Duration: 1 Hour 20 Minutes **Set - A** Full Marks:30

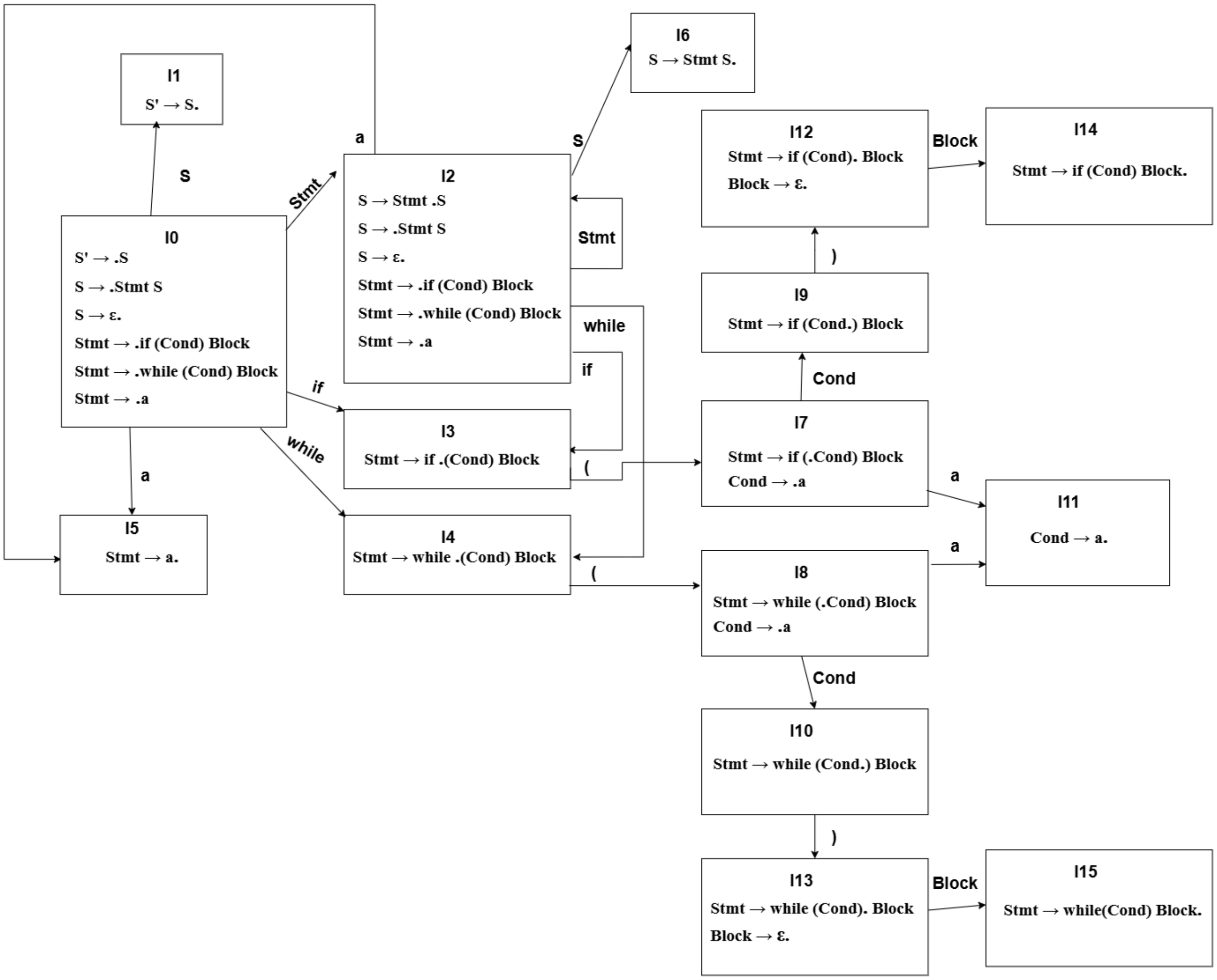
CSE 420: Compiler Design

Figures in the right margin indicate marks.

Answer all the questions

COs	Questions	Marks
CO1	1. What is the goal of compiler error recovery during syntax analysis? Explain the different error recovery mechanisms the compiler can apply to tackle parsing errors in the source program	5
CO1	2. Explain how symbol tables are used to determine what variable comes from what memory location during translating expressions and statements.	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			
	while	if	(	a	)	$	S	Stmt	Block	Cond
0		S3		S5			1			
1										
2		S3		S5				2		
3			S7							
4			S8							
5	R5	R5				R5				
6						R1				
7				S11						9
8				S11						10
9					S12					
10					S13					
11					R7					
12									14	
13									15	
14	R3			R3		R3				
15	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: **if (a) while (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 → 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 → T {E.node = T.node;}
	T → 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 → 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 → F {T.node = new Node(); childList = new List();
	childList.add(F.node); T.node.children = childList;}
	F → id {F.node = new Leaf(id.lexeme);}
	F → (E) {F.node = E.node;}
		10