CS31003 Compilers
Class Test 1
28-Aug-2025, 6:00pm–7:00pm
Maximum Marks: 50

Roll No: _____

Name: _____

[*Write your answers in the question paper itself. Be brief and precise.*]

1. [LL(1) Parsing] Consider the unambiguous if-else grammar.

statement → matched | open
matched → if cond matched else matched | unconditional
open → if cond matched else open | if cond statement

Let us use the short-cut names S (statement), M (matched), O (open), i (if cond), e (else), and u (unconditional), where the upper-case letters stand for non-terminals, and the lower-case letters for terminals. Using left factoring, we restate these rules as follows.

(1) S → M
(2) S → O
(3) M → i M e M
(4) M → u
(5) O → i T
(6) T → M e O
(7) T → S

(a) Write (with justification) the FIRST and FOLLOW functions of the non-terminals. **[8]**

	Value	**Justification**
FIRST(S)	{ i, u }	FIRST(M) U FIRST(O)
FIRST(M)	{ i, u }	FIRST(i M e M) U FIRST(u)
FIRST(O)	{ i }	FIRST(i T)
FIRST(T)	{ i , u }	FIRST(M) U FIRST(S)
FOLLOW(S)	{ $ }	{ $ } U FOLLOW(T)
FOLLOW(M)	{ e, $ }	{ e } U FOLLOW(S)
FOLLOW(O)	{ $ }	FOLLOW(S) U FOLLOW(T)
FOLLOW(T)	{ $ }	FOLLOW(O)

(b) Using the values of Part (a), derive the LL(1) predictive parsing table. **[8]**

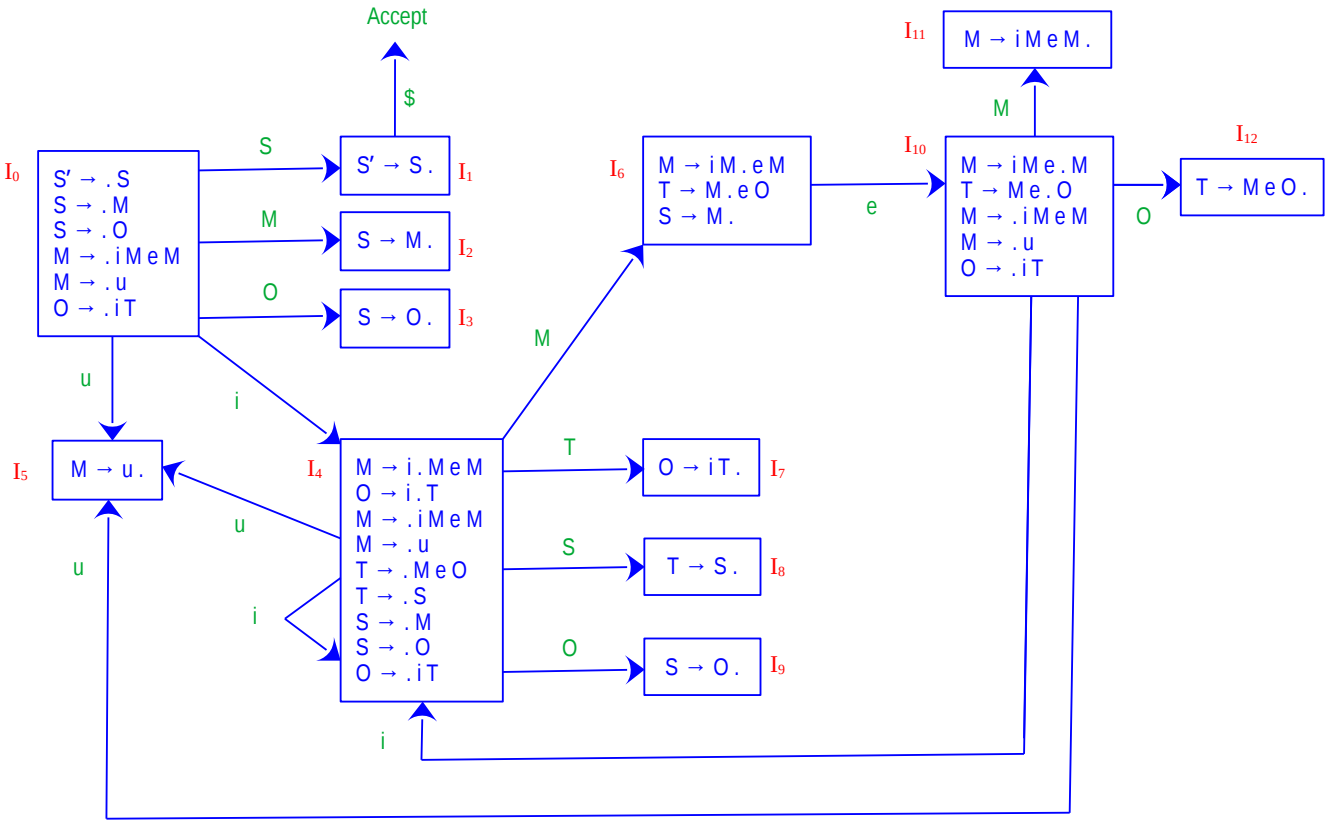
Non-Terminal	**Input symbol (terminal)**			
	i	e	u	$
S	S → M S → O		S → M	
M	M → i M e M		M → u	
O	O → i T			
T	T → M e O T → S		T → M e O T → S	

(c) From the table of Part (b), justify whether the left-factored unambiguous grammar is LL(1). **[2]**

There are three cells in the LL(1) parsing table, that contain multiple entries. So the grammar is not LL(1).

2. [SLR(1) Parsing] Let us continue to use the left-factored grammar of Exercise 1.

(a) Draw the LR(0) automaton for this grammar in the space below. **[12]**



(b) From your automaton of Part (a), derive the SLR(1) (simple LR(1)) parsing table, in the space below. Number the productions as given in Exercise 1. **[10]**

State	ACTION				GOTO			
	i	e	u	$	S	M	O	T
0	s4		s5		1	2	3	
1				Accept				
2				r1				
3				r2				
4	s4		s5		8	6	9	7
5		r4		r4				
6		s10		r1				
7				r5				
8				r7				
9				r2				
10	s4		s5			11	12	
11		r3		r3				
12				r6				

(c) Justify from the parsing table of Part (b) whether the grammar in question is SLR(1). **[2]**

Since no cell in the SLR(1) parsing table contains multiple entries, the grammar in question is SLR(1).

(d) From the SLR(1) parsing table, work out how the string

```
i i u e u
```

is accepted. Show (in the format of the following table) the init step, and all shift and all reduction steps along with the evolution of the stack of states (no need to show the grammar symbols in the stack). For each shift step, write (in the middle column of the table) which symbol is shifted (like `Shift u`), whereas for each reduction step, write the exact production used in the reduction (like `Reduce by O → i T`). **[8]**

Parse stack (stack of states)	Step (Shift / Reduce)	Input left to be consumed
0	Init	i i u e u $
0 4	Shift i	i u e u $
0 4 4	Shift i	u e u $
0 4 4 5	Shift u	e u $
0 4 4 6	Reduce by M → u	e u $
0 4 4 6 10	Shift e	u $
0 4 4 6 10 5	Shift u	$
0 4 4 6 10 11	Reduce M → u	$
0 4 6	Reduce by M → i M e M	$
0 4 8	Reduce by S → M	$
0 4 7	Reduce by T → S	$
0 3	Reduce by O → i T	$
0 1	Reduce by S → O	$
0 1	Accept	$