Question 1 .....

Given are the following two syntactically different regular expressions:

$R1 := ((a|b)^*.c^*.(a^*|b^*)^*)^*$
$R2 := (a^*|b^*)^*.(c|(a|b)^*)^*$

Investigate by way of the DFA$_{min}$ method whether $L(R1) = L(R2)$. ..... **[2 Points]**

Question 2 .....

Given is the somewhat "ambiguous" English *sentence*

s := **Susan opens the file with the password**

Construct a *small* context-free grammar G (with only a few rules) for "very simple English", such that s is in $L(G)$, and demonstrate with the help of s that your grammar G is an *ambiguous* grammar.

Advice: *Make sure to give meaningful grammatical names to the Non-Terminal symbols* of G (e.g.: Sentence, Subject, Object, Predicate, Noun, Verb, or similar *linguistic* terms), such that the exam's assessor (marker) is easily convinced about s being in $L(G)$.

..... **[4 Points]**

Question 3 .....

Apply the Look-Ahead (LA)-Set method to investigate whether the following grammar is suitable for LL1 parsing (whereby any blank_spaces are to be ignored as irrelevant, and **bold** symbols shall be regarded as **terminal** symbols):

COMMAND	::= **c**
COMMAND	::= IFTHENELSE
IFTHENELSE	::= **i** BOOLEAN **t** COMMAND ALTERNATIVE
ALTERNATIVE	::= **e** COMMAND
ALTERNATIVE	::= *// epsilon, nothing*
BOOLEAN	::= **b**

..... **[2 Points]**

Question 4 .....

Let **L***dyn* be some programming language with *dynamic scoping* for function-calls. Explain briefly,

a) *which data structure* will be best for the implementation of the *symbol table* in the compiler of **L***dyn*,
b) and *why* that is so.

..... **[2 Points]**

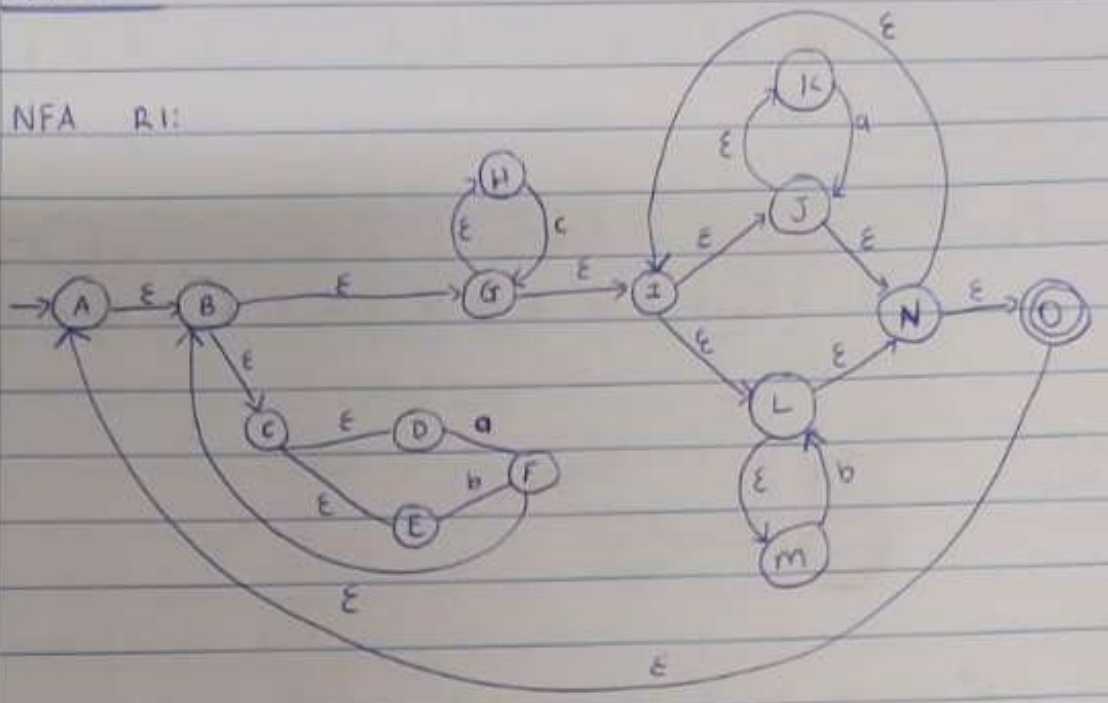
Question 5 .....

a) With reference to Theoretical Computer Science, explain briefly why a *static* type-checker cannot be both correct and complete.
b) What is the *practical consequence* of this fundamental theorem?

There are no further questions.

Question 1 (5)

NFA R1:



NFA to DFA

S_0: {A, B, C, D, E, G, H, I, J, K, L, M, N, O} with ε

move(S_0, a) = ε closure ({F, J})

= {F, J, A, B, C, D, E, G, H, I, K, L, M, N, O}

= S_1

move(S_0, b) = ε closure ({F, L})

= {F, L, A, B, C, D, F, G, H, I, J, K, M, N, O}

= S1

move(S_0, c) = ε-closure ({G})

= {G, A, B, C, D, E, H, I, J, K, L, M, N, O}

= S_0

move(S_1, a) = ε closure ({F, J})

= {F, J, A, B, C, D, E, F, G, H, I, K, L, M, N, O}

= S1.

Seen

move({S1}, b) = ε-closure({F, M})

= {F, M, A, B, C, D, E, F, G, H, I, J, K, L, N, O}

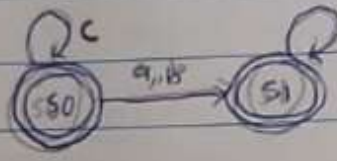
= S1

move(S1, c) = ε-closure({G})

= {C, A, B, D, E, FF, G, H, I, J, K, L, M, N, O}

= S1S1

DFA :

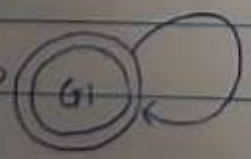


min DFA :

G1 : {S0, S1} Both are accepting, so we don't have another division.

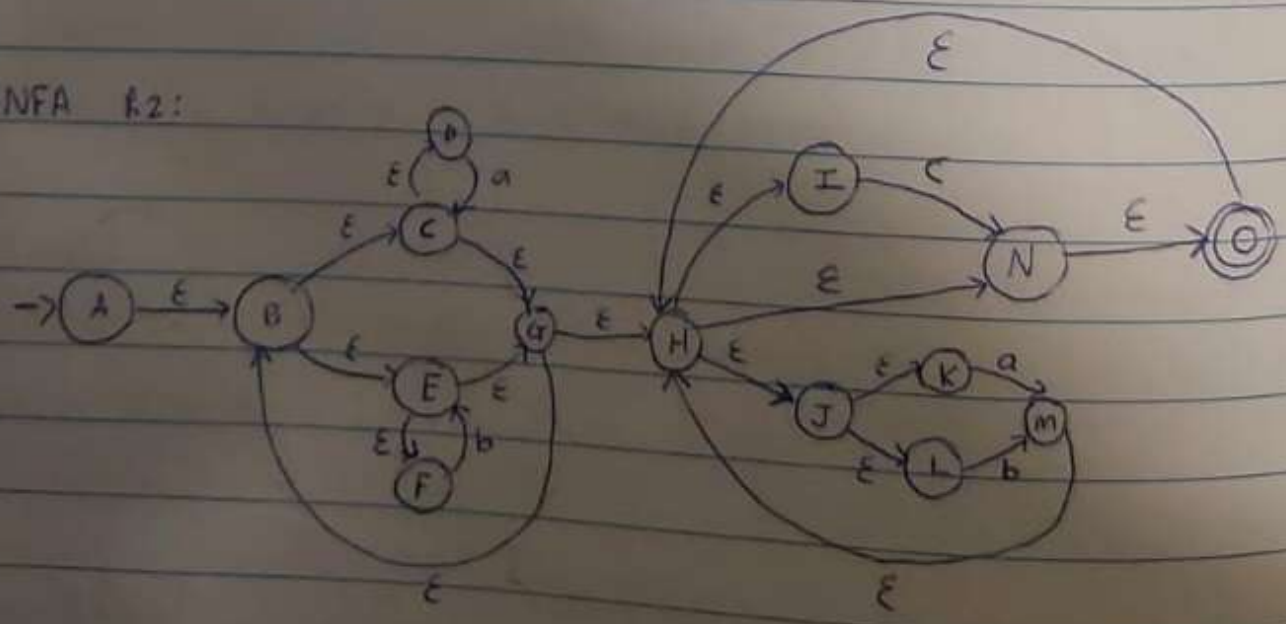
G1	a	b	c
S0	G1	G1	G1
S1	G1	G1	G1

∴ G1 is maximally consistent.

∴ 

} correct ✓

NFA k2:



★ NFA to DFA:

S_0 : { A, B, C, D, E, F, G, H, I, J, K, L, N, O }

move(S_0, a) = ε closure ({C, M})

= {C, M, A, B, D, E, F, G, H, I, J, K, L, N, O}

= S1

move (S_0, b) = ε closure ({E, M})

= {E, M, A, B, C, D, F, G, H, I, J, K, L, N, O}

= S1

move (S_0, c) = ε closure ({N})

= {N, A, B, C, D, E, F, G, H, I, J, K, L, N, O}

= S0

move (S_1, a) = ε closure ({C, M})

= S1

move (S_1, b) = ε closure ({E, M})

= S1

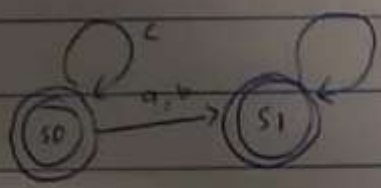
feel-

move (S_1, c) = ε closure ({N})

= {N, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O}

= S1

∴ DFA :



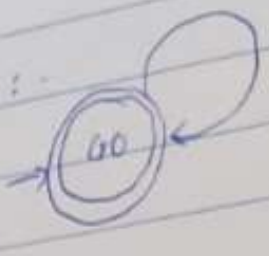
Min DFA : - G0: { S_0, S_1 (both are accepting. thus 1 group)

G0	a	b	c
S0	G1	G1	G1
S1	G1	G1	G1

∴ G0 is maximally constant and

min DFA PTO →

min DFA :-



correct ✓

G0 = G1

∴ Thus I have proven that the min DFA for both languages are the same, thus $L(R_1) = L(R_2)$ ✓

Question 3.

First (command) = First(c) ∪ First (IF THENELSE)
= {c} ∪ First {i}
= {c, i}.

First (IFTHENELSE) = First (i)
= {i}.

First (Alternative) = First (e) ∪ First ()
= {e, }.

First (Boolean) = First (b)
= {b}

command' ::= command $	{$} ⊆ Follow (command)
Command ::= IFTHENELSE	Follow(comand) ⊆ Follow(IFTHENELSE)
IFTHENELSE ::= ; Bolean	{;} ⊆ Follow (Boolean), First(Alternative) ⊆ Follow(command),
	Follow(IFTHENELSE) ⊆ Follow (Alternative)
ALTERNATIVE ::= e Command	Follow (ALTERNATIVE) ⊆ Follow(COMMAND)

∴ Follow (COMMAND) = {$, e}
Follow (IFTHENELSE) = {$, e}
Follow (BOOLEAN) = {t}
Follow (ALTERNATIVE) = {$, e}

PTO.

la (Command := c)	= First(c) = {c}
la ((Command := IfthenElse)	= First (IfthenElse) = {i}
la (IfthenElse := i Bool...)	= First (i) = {i}
la (Alternative) := Command)	= First (e) = {e}
la (Alternative :=)	= First() ∪ Follow (Alternative) = {}
la (Boolean := b)	= First(b) = {b}

Table.

	c	i	t	e	b
Command.	Command := C	Command := IFTHENELSE			
IFTHENELSE		IFTHENELSE := i Boolean then Command else Alternative			
Alternative.				Alternative := Command. Alternative :=	
Boolean.					Boolean := b

This is not an LL(1) parser because 2 productions, share the same first. Thus we are unable to uniquely identify a production.

Wrong? Conclusion from wrong LA-Set

Question 4

a) Stack ✓

b. If the symbol table is implemented as a stack of scopes, it would make dynamic scoping a lot easier to deal with. For example, if a variable "x" occurs in more than one scope, it is always bound to "current" scope (on top of the stack). ✓

Quest

Question 5

a. To "accept" an arbitary non-terminating program that will not run into any type error during its run time, the type checker would need to "know" for sure whether the program is a halting or a non halting program, however the halting program is undecidable.

b. The static type checker will reject atleast one type-error-free input program.