Note (i) Write your answers neatly and precisely in the space provided with with each question including back of the sheet. You won't get a second chance to explain what you have written.
(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.

1. Given the grammar

$$S \to AA$$
$$A \to AAA|a|bA|Ab$$

is it true that it generates all strings with even number of a's ? Either give a counterexample or state a formal induction assertion for S and A (induction proof not required) that makes the claim true.

Consider a string 'b'. It contains 0 number of a's.
The terminal in which A ends ~~ab~~ is always an 'a'. So,
Grammar can't generate 'b' and hence it does not generate
all strings with even no. of a's.

1

Basically $S \to AA$ and $A \to a$ makes the grammar such that
there have to be minimum 2 a's in the strings gen

This is only 1 exception

2. Let L be a language consisting of all strings over $\{a, b, c, d\}$ that do not contain equal number of a,b,c,ds, i.e., aabcc is in the language but abccadbd is not. Is L CFL - Justify your answer.

Yes, L is a CFL. ✓

Note (i) Write your answers neatly and precisely in the space provided with with each question including back of the sheet. You won't get a second chance to explain what you have written.
(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.

1. Let $L_H = \{< M, w > | M \text{ halts on input } w\}$. Is L_H
   (i) recurisive (ii) r.e. but not recursive (iii) not r.e. ? Justify. **(10)**

~~L_H is the language of H~~

L_H is r.e but not recursive.

10

<u>Argument for r.e</u> : L_H is r.e. because we can design a turing machine which will stop on all instances of yes of L_H. ~~The T.M will~~ The T.M. of L_H will take in the input $\langle M, w \rangle$. It will run w on M. If $\langle M, w \rangle \in L_H$, on running w on M, M will halt and if it halts, we will output yes. So this T.M for L_H will always output yes if $\langle M, w \rangle \in L_H$. Hence this is r.e.

<u>Argument for not recursive</u>: We can prove this by contradiction

In class, we have done that L_u is not recursive where $L_u = \{\langle M, w \rangle \mid M \text{ accepts } w\}$.

Else if L_u is recursive $\Rightarrow$ L_u is re & $\boxed{\bar{L_u} \text{ is r e}}$ $\rightarrow$ Not true

Hence, L_u is not recursive.

$L_u \leq_g L_H$ where g is a turing computable function $g(M, w) = \langle M' \rangle$ s.t. if w is accepted by M, M' will halt on Σ^* else M' will never halt on input Σ^*

Now suppose L_H is Recursive,

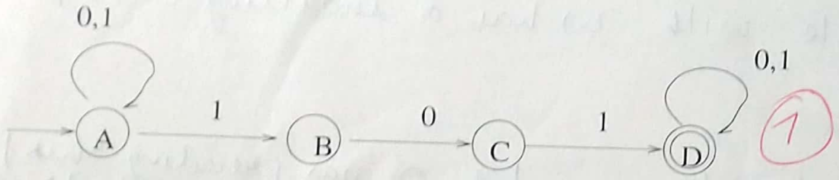
For no instances of L_H, we can design a TM which will output NO

can design a T.M for L_u which always halt

L_u is recursive, contradiction $\Rightarrow$ L_H is not recursive

2. A student unconvinced by the diagonalization argument for proving L_d is not r.e., approaches her Professor with the following doubt. Since the set L_d is dependent on the ordering of strings, what if a different ordering $\mathcal{O}'$ is used ? Why will the previous L_d still continue to be a non r.e. set although it does not correspond to the diagonal in $\mathcal{O}'$? Can you answer her doubts ? You can assume that both orderings can be computed using a TM. **(10)**

Note (i) Write your answers neatly and precisely in the space provided with with each question including back of the sheet. You won't get a second chance to explain what you have written.
(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.



7

1. (i) What is the language accepted by the above finite automaton ? Answer in 1 line - no proof needed.

3 A string with '101' as a substring.

$L = \{ w \in \{0,1\}^* \mid 101 \text{ is a substring of } w \}$

(ii) Suppose we flip the accepting and the non-accepting states - then what is the language accepted by the modified finite automaton ?

$L = \{ w \in \{0,1\}^* \mid$ ~~101 is not a substring of w~~ $\}$ w ending in '1' or '10'

Every string is accepted

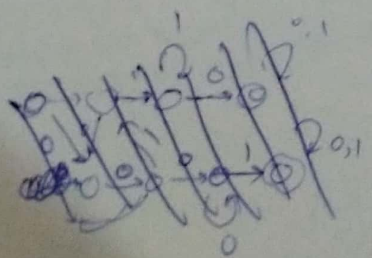
(iii) What can you conclude from this ? (1 sentence).

We can conclude that for every NFA, there exist ~~the~~ an DFA that accepts converse strings of original NFA.

or

4 → For every machine M, there exists a converse machine M' if M is DFA. (Doesn't hold with NFA)

2. Design a DFA for the language $L = \{w \in \{0,1\}^* | w$ contains the substring 110 or 0001 $\}$. For example 01010101 $\notin L$ but 01110000101 $\in L$. (10)



Note (i) Write your answers neatly and precisely in the space provided with with each question including back of the sheet. You won't get a second chance to explain what you have written.
(ii) You can quote any result covered in the lectures without proof but any other claim should be formally justified.

1. Let $L_H = \{<M, w> \mid M$ halts on input $w\}$. Is L_H
(i) recurisive (ii) r.e. but not recursive (iii) not r.e. ? Justify. (10)

0

Ans L_H is recursive.

I can construct a machine M_H as:-

1) | M w |
2) | w_1 --- w_n | ← copy w on this tape from above.
3) | |
4) | q_i | ← set of all final states of M saying Yes
5) | q_F | ← Final states of M saying No

Now M_H will simulate M on w as, it will have an input tape with M & w on it. It will copy w on tape 2. It will write the states of M on tape 4 and use tape for its own processing.

Now M on tape is actually the δ of M. So M_H will simulate δ of M on tape 2 (w). If M says Yes i.e. it stops in a final state which can be checked from Tape 4 our M_H transits to its final state & says Yes. If M halts on final state i.e. q_i ∈ Tape 5, (q_i on Tape 5) M halts on a final state saying $w \notin M$, My M_H returns No by transitioning into a final state corresponding to No

2. A student unconvinced by the diagonalization argument for proving L_d is not r.e., approaches her Professor with the following doubt. Since the set L_d is dependent on the ordering of strings, what if, a different ordering O' is used ? Why will the previous L_d still continue to be a non r.e. set although it does not correspond to the diagonal in O' ? Can you answer her doubts ? You can assume that both orderings can be computed using a TM. (10)