National University of Computer and Emerging Sciences
Karachi Campus

Theory of Automata (CS3005)

Date: May 28th 2024

Course Instructor(s)

Dr. M. Shahzad, Dr. Nasir-uddin, Mr. Syed Faisal, Ms. Shaharbano, Ms. Bakhtawar, Ms. Zain Noreen

Final Exam

Total Time (Hrs): 3
Total Marks: 50
Total Questions: 4

SOLUTION PAPER

_____________ Roll No   ____________ Section   _________________________ Student Signature

Do not write below this line

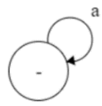
Attempt all the questions.

CLO 2: Prove properties of languages and automata with rigorously formal mathematical methods

Q1: Regular Languages **[10 marks]**

1. Let D1 and D2 be two DFAs with n states each, accepting languages L1 and L2 respectively. The resulting DFA after intersecting D1 and D2 has:
   a) At most n states
   b) At most 2n states
   c) At most n^2 states
   d) At most 2^n states

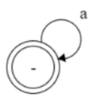
2. What language is generated by the given DFA?
   a) Accepting strings having null only
   b) Accepting strings having any number of a's and null
   c) Accepting all strings upon a's only
   d) None of the strings



3. How is the complement of a DFA constructed?
   a) By swapping the accepting and non-accepting states.
   b) By swapping the initial state with the accepting state.
   c) By removing all accepting states and changing the initial state into accepting state.
   d) By swapping the accepting state with the initial state and changing the directions of the transitions.

4. What is the ε-closure of a state in an NFA-ε?
   a) The set of states that can be reached from the state on a single transition.
   b) The set of states that can be reached from the state without consuming any input symbols.
   c) The set of states that can be reached from the state on any input symbol.
   d) The set of states that can be reached from the state after consuming all input symbols.

5. What is the compliment of the given DFA over Σ = {a}?
   a) The same language
   b) Accepting only epsilon
   c) The Null set
   d) Not possible



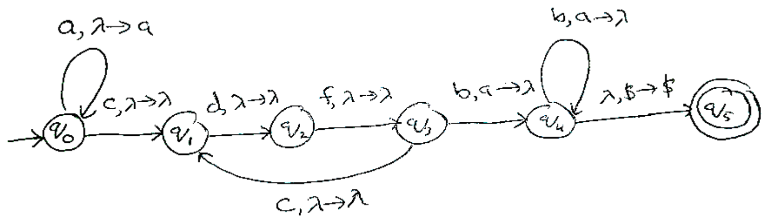
National University of Computer and Emerging Sciences

Karachi Campus

CLO 5: Define Turing machines, PDA machines performing simple tasks.

Q2: Push Down Automata (PDA) **[10 marks]**

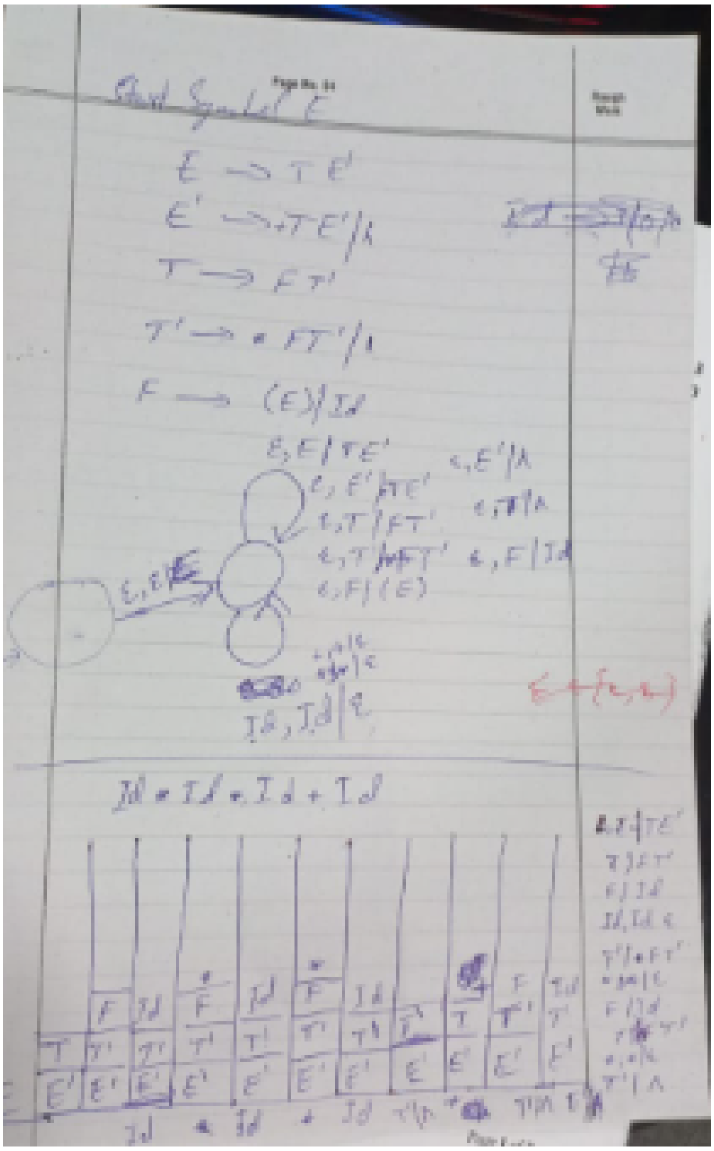
a) **[5 marks]** Design PDA such that it accepts the $L = \{a^n(cdf)^+b^n : n \geq 1\}$



b) **[5 marks]** Convert the following context free grammar to a PDA that accepts the same language. Show tape configuration and stack contents for the input ***id*id*id+id***.

E → TE' E' → +TE' |λ T → FT'	T'→ *FT' |λ F→ (E) |id

Note: E, T, F, E' and T' are variables, whereas +, *, (and) are terminals



CLO 3: Design of automata, RE and CFG

Q3: Context Free Grammar (CFG)	**[15 marks]**
a) **[5 marks]** Simplify and clean the given following grammar in proper order and then find CNF: S → ABCD| AB|CD A → abd | bdc| EF B → xyz | wxy |xE | yF C → xA | yB | wF E → a | b | c	b) Consider the following grammar and use the given string to ensure whether the grammar is ambiguous or not: *<sentence> → <subject><predicate>* *<subject> → noun | article noun | <object>* *<predicate> → verb <object>* *<object> → noun | article noun* **String to parse**: Ahmed hits the ball

No null and unit productions

Removing Useless symbols

D&F are non-generating

S → AB
A → abd | bdc
B → xyz | wxy |xE
C → xA | yB
E → a | b | c

Now C is unreachable
remove C

S → AB
A → abd | bdc
B → xyz | wxy |xE
E → a | b | c

Now Converting into CNF

S→AB

A→ $T_a T_b T_d$| $T_b T_d T_c$

B→ $T_x T_y T_z$ |$T_w T_x T_y$| $T_x E$

E → a | b | c

T_a →a

T_b→b

T_c→c

T_d→d

T_x→x

T_y→y

T_z→z

T_w→w

S→AB

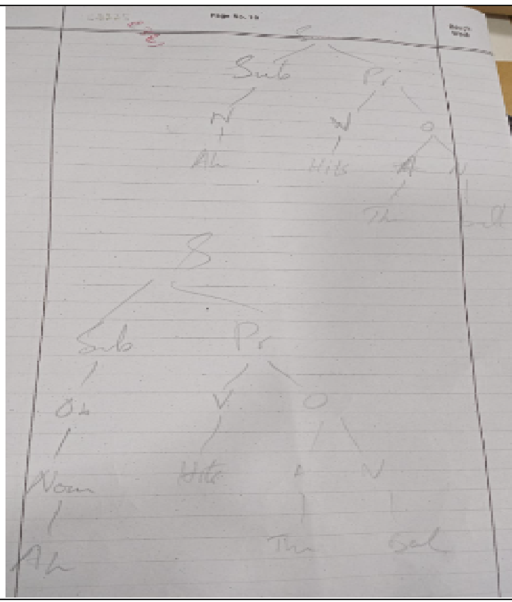
A→ $T_a N$ | $N T_c$

N → $T_a T_d$

B→$M T_z$ | $T_w M$ | T_x E

M→$T_x T_y$

E→ a | b | c



c) **[5 marks]** For the terminals set T = {a, c, d, e, #} design a grammar G that satisfies the following conditions:
- L(G) is a set of palindromes.
- All the strings in L(G) start and end with terminal "#".
- The terminal at the mid position is always "e".
- Terminal "a" is always followed by terminal "b".

Hint: Few elements of the language L(G) = {#e#, #eee#, #abeab#, #dcabeabcd# ......}

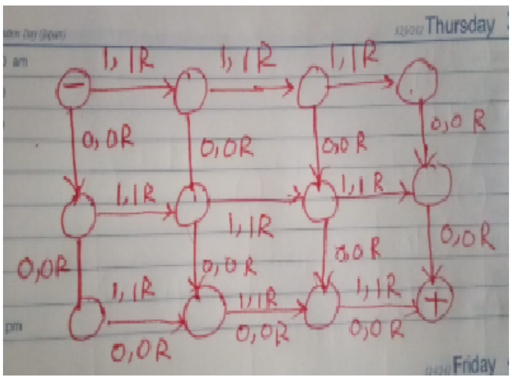
S→#A#
A→e|B|
B→aCa|cAc|eAe|dAd
C→bAb

CLO 5: Define Turing machines, PDA machines performing simple tasks.

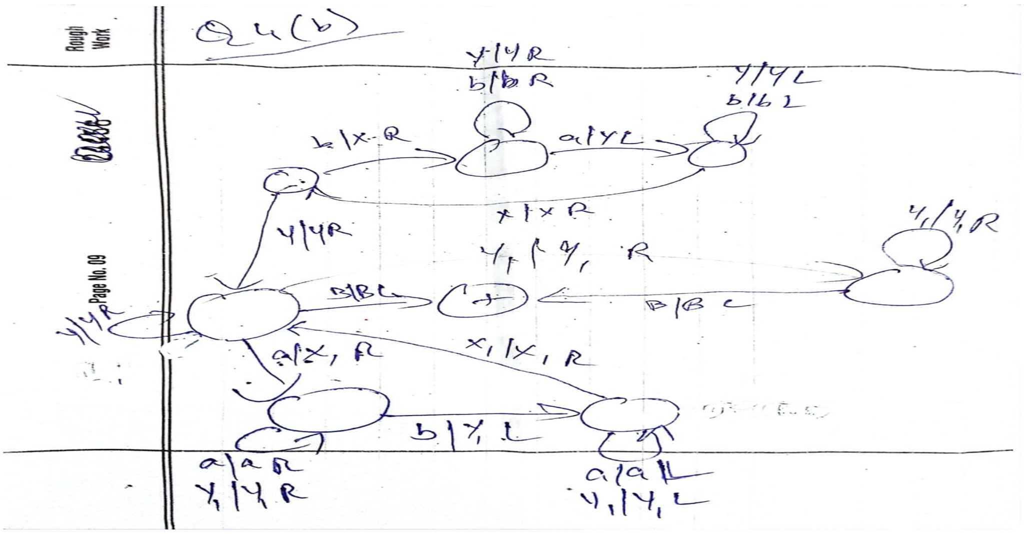
Q4: Turing Machines (TM) **[15 marks]**

Design TM for the following descriptions:

a) **[2.5 marks]** The set of all strings of length 5, defined over Σ = {0,1} contains at least two 0's:
Some accepted strings: 00000, 00001, 00111, 10101
Some rejected string: 11110, 11111



b) **[2.5 marks]** L = {$b^m a^n a^m b^n$ | m, n > =0}



c) **[5 marks]** The TM that processes a number 'x' and performs different actions based on following specific conditions:

1. It first checks if the number 'x' is an even number.
2. If 'x' is even, it then checks if 'x' is divisible by 3:
3. If 'x' is even and divisible by 3 (e.g., 6, 12, 18, ...), display 'x' number of 1's on the tape.
4. If 'x' is not even but is divisible by 3 (e.g., 3, 9, 15, ...), display 'x' number of 0's on the tape.
5. If 'x' is not divisible by 3, display all blanks on the tape.

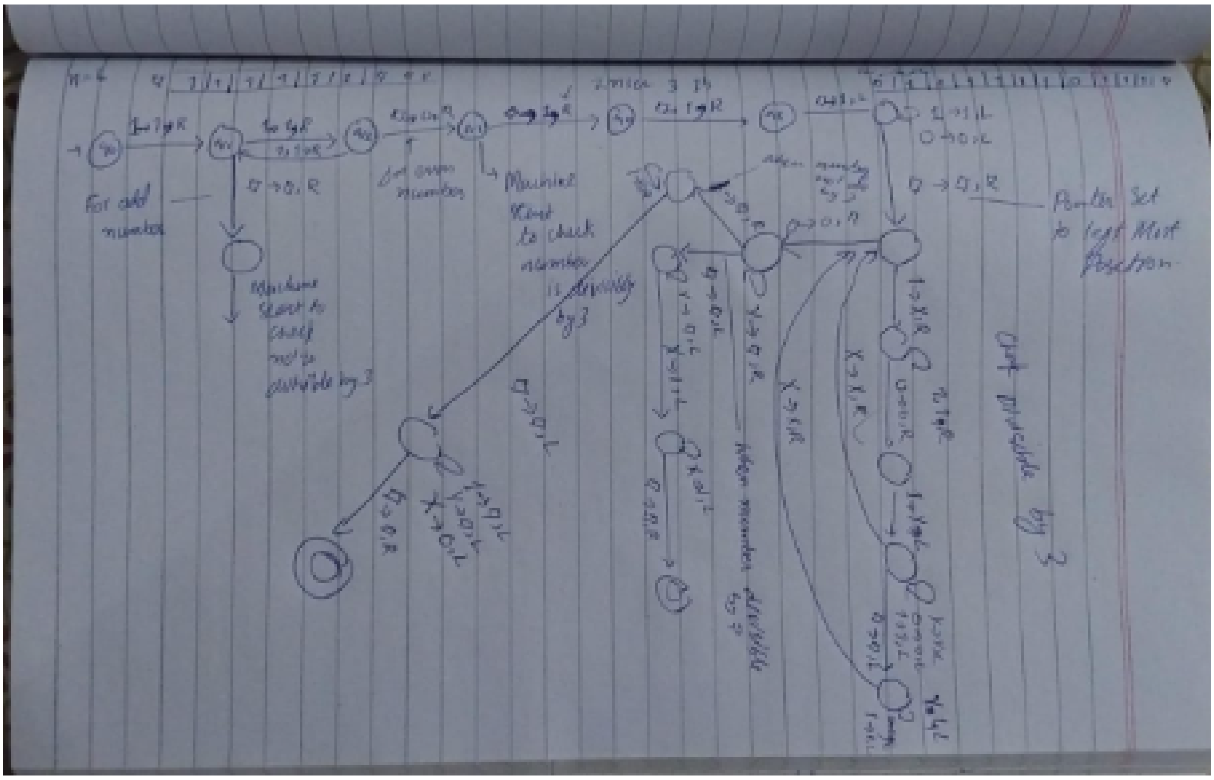
Few input/output examples:

If x=6 then Input -> ♦♦♦♦♦111111♦♦♦♦♦ output - > ♦♦♦♦♦111111♦♦♦♦♦

If x=9 then Input -> ♦♦♦♦♦111111111♦♦♦♦♦ output - > ♦♦♦♦♦000000000♦♦♦♦♦

If x=5then Input -> ♦♦♦♦♦11111♦♦♦♦♦ output - >♦♦♦♦♦

Note: Use Unary Representation for ‘x’.



d) **[5 marks]** Suppose that we have two binary integers on a tape, separated by a symbol 'c'. Given an algorithm and a two tape Turing Machine to compute the sum of those two integers. This machine will compute the sum and put it onto the first tape:

For example, given the input tape:

1	1	C	1	1	0
□	□	□	□	□	□

we will split the two inputs onto two tapes like this.

□	1	1	0	□
□	1	1	□	□

The output will be put onto the 1st tape:

1	0	0	1	□
(don't care)				

Hint: You should start by copying the first string/number from tape 1 to tape 2, then position the heads of both tapes on the right-most symbol of each string. After that, we can enact a binary addition, working right to left, in much the way that you would do manually.

Note: You may design any variant/combination of variants of TM or simple TM to express this problem.

Algorithm:

1. Start by copying the 1st number to tape 2.
2. let's shift the tape head on tape 1 to the end of the 2nd number, and then position both tape heads on the right-most digits.
3. If we see two zeros, we write a zero onto the answer in tape 1. If we see a one and a zero, we write a one onto the answer in tape 1. Either way, we then shift the tape heads left to the next higher pair of digits. Of course, that leaves the case of seeing a pair of ones. For that, we would write a '0' and "carry the one".
4. So we will use state q_2 to add digits with no carry, and a different state, q_3, will enact the rules for addition when we have a carry. Here we have added the "with

carry” addition rules, including the case of adding 0+0 with a carry, which gives us a sum of 1 but returns us to the non-carry state q_2.

5. Now it’s time to start thinking about how to end this.
   a. If we are in state 2 (no carry), and looking at two empty cells, then we have finished adding the numbers and can stop.
   b. If we are in state 3 (carry), and looking at two empty cells, then we would need to write the 1 from the carry onto the left of the answer.
   c. If we are in either of those states and see only one empty cell, that means that one of our two numbers has more digits than the other. We would treat those empty cells as if they contained a zero.
6. End