~~35/40~~

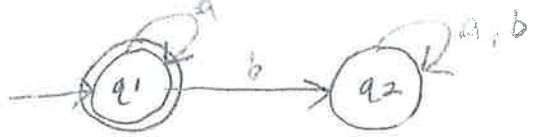
1-4. *Finite automata.* Let us consider the following finite automaton (FA) for recognizing words in the alphabet {a,b} that only have a's (and do not have any b's). This automaton has two states: the start state which is also final, and the sink state. From the start state, letter 'a' leads back to the same state, while letter 'b' leads to the sink state.

10/10 1. On the example of the word aaa accepted by the given automaton, show how this word will be represented as xyz according to the pumping lemma for FA. For this sequence, check -- by tracing step-by-step -- that the sequence xy^2z is indeed accepted by this automaton.

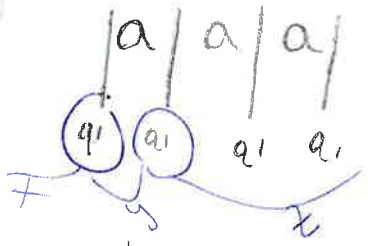
10/10 2. Use the general algorithm to transform the given FA into a context-free grammar (CFG). Show, step-by-step, how the word aaa can be derived in this CFG.

10/10 3. Use the general algorithm to transform the given FA into a Turing machine (TM). Show, step-by-step, how this TM will accept the word aaa.

~~9~~/10 4. Use the Pumping Lemma for FA to prove that the language $\{p^n q^n\} = \{\Lambda, pq, ppqq, pppqqq, ...\}$ cannot be recognized by a finite automaton.

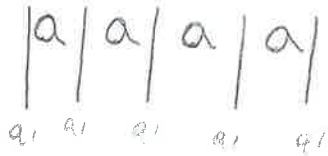


1)



x = ε
y = a
z = aa

x y y z = a a a a



the word aaa is accepted by the automata after tested with pumping lemma

2) $Q_1 \to aQ_1$

$Q_1 \to bQ_2$

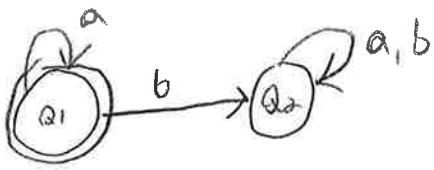
$Q_1 \to \varepsilon$

$Q_2 \to aQ_2$

$Q_2 \to bQ_2$

$Q_1 \to aQ_1 \to aaQ_1 \to aaaQ_1 \to aa$

3)



start, ␣ → R, Q1

Q1, ␣ → accept

Q1, a → R,

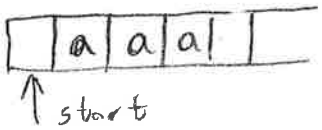
Q1, b → R, Q2

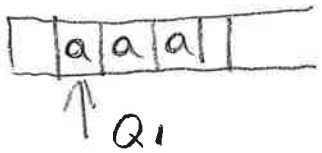
Q2, ␣ → reject

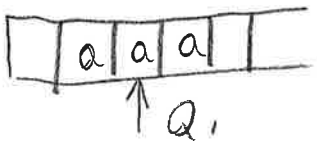
Q2, a → R

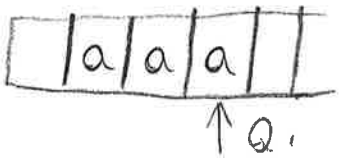
Q2, b → R

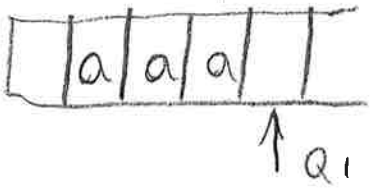
a a a

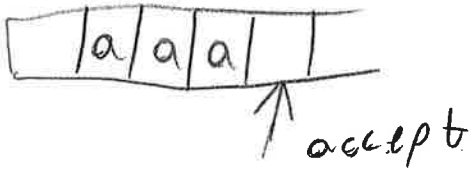












④ $\{p^n q^n\} = \{\lambda, pq, ppqq, pppqqq, \ldots\}$

$L = \{\lambda, pq, ppqq, pppqqq, \ldots\}$

Proof by contradiction, let's assume that L is regular. Then by pumping lemma, there exists p such that every word from L whose length is $\geq p$ can be represented as xyz, where:

$len(y) > 0$, $len(xy) \leq p$ and for all y $xy^i z$ is in L

Let's take

$p^p p^p p^p q^p q^p q^p$

p...p p...p p...p q...q q...q q...q

p times p p p p p p

$len = 6p > p$

$len(xy) \leq p$

so y has only p's

xyyz → p...p p...p p...p q...q q...q q...q

↖ we have $> p$ p's since we added y

This is no longer same word repeated 3 times not in L

39 ~~40~~/40

5-8. *Pushdown automata (PDA) and context-free grammars (CFG).* The following CFG generates the language $\{p^n q^n\}$: it has the starting variable S and the rules $S \to \varepsilon$, $S \to pSq$.

10/10 5. Show, step-by-step, how the word ppqq will be generated by this CFG.

10/10 6. Use the general algorithm to generate a PDA that recognizes the words generated by this CFG -- and only these words. Show, step-by-step, how the resulting PDA will accept the word ppqq.

~~9~~/10 7. Use the general algorithm to transform the given CFG into Chomsky normal form.

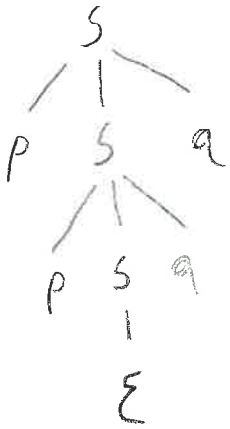
10/10 8. On the example of the word ppqq generated by the given CFG, show how this word will be represented as uvxyz according to the pumping lemma for CFG. For this sequence, check -- by tracing step-by-step -- that the sequence uv^2xy^2z is indeed generated by this CFG.

$S \to \varepsilon$

$S \to pSq$

ppqq

5)

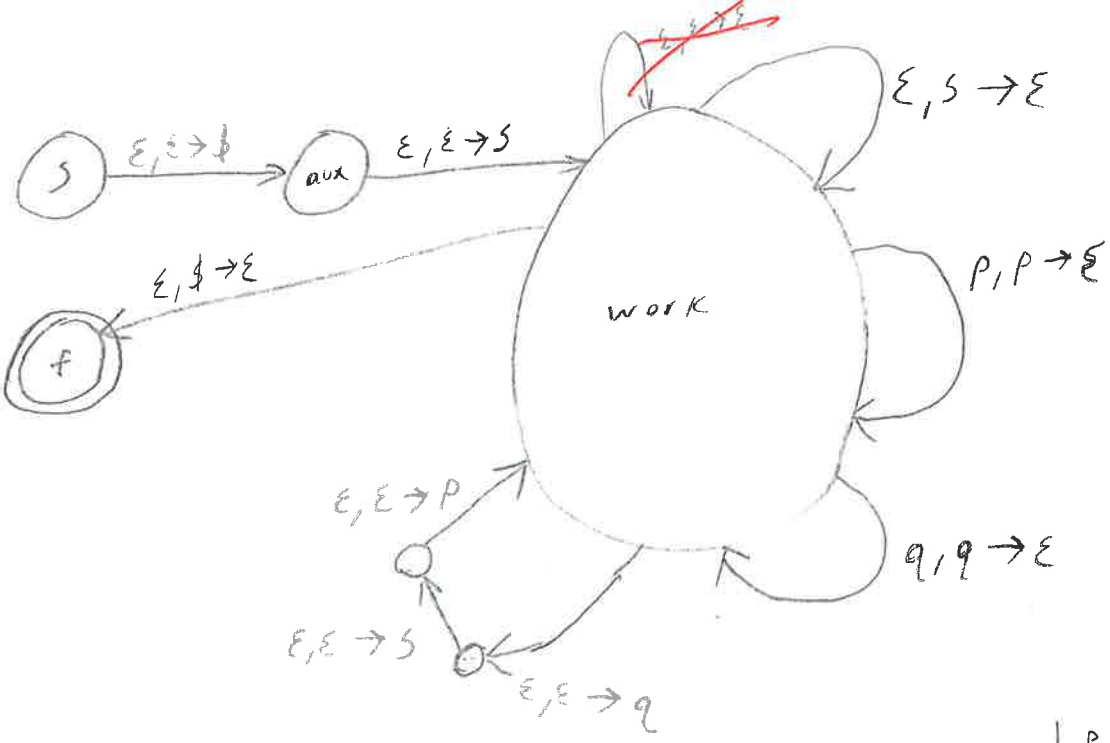


$S \to pSq \to ppSqq \to ppqq$

6)

$S \to \varepsilon$

$S \to pSq$



ppqq

(7) CFG → CNF

step 0

(1) $S \to \varepsilon$

(2) ~~$S \to pSq$~~

aux (3) ~~$S_0 \to S$~~

(4) $S \to pq$

step 1

(5) $S_0 \to \varepsilon$

(6) ~~$S_0 \to pSq$~~

(7) $S_0 \to pq$

step 2

(8) $S \to V_{pS}q$

(9) $V_{pS} \to pV_S$

(10) $S \to V_pV_q$

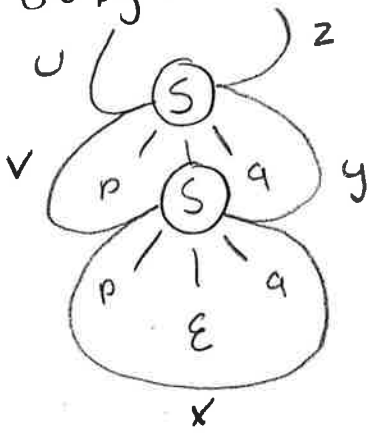
(11) $S_0 \to V_{pS_0}q$

(12) $V_{pS_0} \to pV_{S_0}$

(13) $S_0 \to V_pV_q$

---

(8) $uvxyz = \wedge ppqq\wedge$



$u = \wedge$

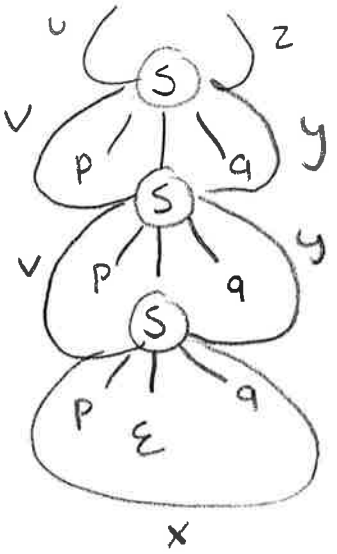
$v = p$

$x = pq$

$y = q$

$z = \wedge$

$uv^2xy^2z = \wedge pppqqq\wedge$ ✓

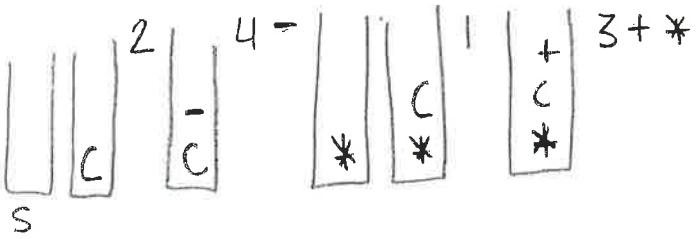


10/10

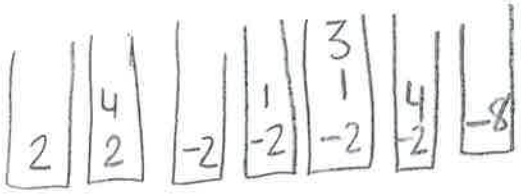
9. Use the general stack-based algorithms to show:

-2 4

- how the compiler will transform the expression (2 − 4) * (1 + 3) into postfix form, and
- how it will compute the value of the resulting postfix expression.



= 24-13+*



= -8

10/10

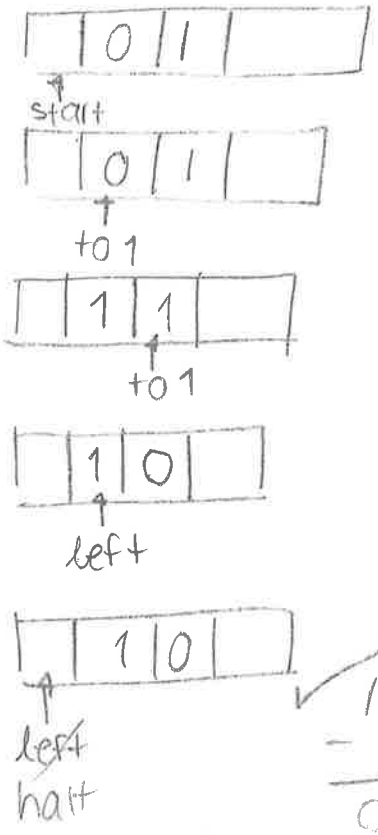
10. Design a Turing machine (TM) for subtracting 1 from the input n in binary code; assume that the input is non-zero. Trace, step-by-step, how your TM will perform when n = 10_2 (i.e., when n is equal to number 2 as described in the binary code).

11/1

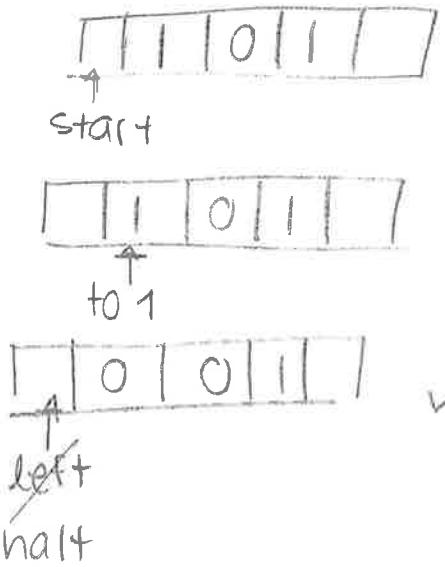
1001

start, ___ → R, to 1
to 1, 1 → 0, L, left
to 1, 0 → 1, R

left, 1 → L
left, blank → halt



10
- 1
01



101
- 1
100