AMERICAN UNIVERSITY OF SCIENCE AND TECHNOLGY

Final project

Fall 2022-2023

Theory of Computation

CSI 301

Yaacoub Khalil

12200055

**Exercise # 1**

a) { w ∈ {0, 1}\* | w contains at least three 1s }

Answer: G = (V, Σ, R, S) with set of variables V = {S, X}, where S is the

start variable; set of terminals Σ = {0, 1}; and rules

S → X1X1X1X

X → 0X | 1X | ε

b) { w ∈ {0, 1}\* | w = wR and |w| is even }

Answer: G = (V, Σ, R, S) with set of variables V = {S}, where S is the start

variable; set of terminals Σ = {0, 1}; and rules

S → 0S0 | 1S1 | ε

c) { w ∈ {0, 1}\* | the length of w is odd and the middle symbol is 0 }

Answer: G = (V, Σ, R, S) with set of variables V = {S}, where S is the start

variable; set of terminals Σ = {0, 1}; and rules

S → 0S0 | 0S1 | 1S0 | 1S1 | 0

d) { ai bj ck | i, j, k ≥ 0, and i = j or i = k }

Answer: G = (V, Σ, R, S) with set of variables V = {S, W, X, Y, Z}, where

S is the start variable; set of terminals Σ = {a, b, c}; and rules

S → XY | W

X → aXb | ε

Y → cY | ε

W → aW c | Z

Z → bZ | ε

**Exercise # 2**

Convert the following CFG to Chomsky Normal Form (CNF)

S  a X | Yb

X  S | λ

Y  b Y | λ

Answer:

λ -Production

S 🡪 aX|Yb|a|b

X 🡪 S

Y 🡪bY|b

Unit-Production

S 🡪 aX|Yb|a|b|aS

Y 🡪bY|b

CHOMSKY NORMAL FORM

Ta 🡪a

Tb 🡪 b

S 🡪TaX|YTb|a|b|TaS

Y 🡪 TbY|b

**Exercise # 3**

Is the following grammar ambiguous? Why?

S → aSbS | SbaS | λ

Answer:

S🡺aSbS 🡺aSbaSbS 🡺 abaSbS 🡺ababS 🡺 abab

S 🡺aSbS 🡺abS 🡺abaSbS 🡺ababS 🡺abab

If we draw the tree we will have two different trees that gives the same string abab by that we conclude that

This grammar is ambiguous

**Exercise # 4**

1. Use the pumping lemma to show that {ai b j ck | i < k and j > k} is not a context-free language.

2. Use the pumping lemma for regular sets to show that {wwR | w ∈ (a + b ) \*} is not a regular set.

3. Let L = { anbncndn |n ≥ 1 }. Show that L can be expressed as the intersection of

two context-free languages.

Answer:

1. Assume that the language is context-free when looking for contradiction. L being infinite, the pumping lemma can be used. We select W where W L and |w|= m and assume that m is the pumping lemma integer.

W=uvxyz and |vxy|=m and |vy|>=1, where W=ambm+2cm+1 and uvixyiz L for all i>=0.

Example 1: Vxy is inside a

We would have am+kbm+2cm+1 if we took uv2xy2z, but that doesn't belong to L because the number of a's can be more than the number of b's and c's.

Case 2: Vxy lies inside of B

We would have ambm+2-kcm+1 if we took uv0xy0z, but that doesn't belong to L because the number of b's can be fewer than the number of c's and as.

Case 3: VxY lies inside of C

If we take uv2xy2z, we would have ambm+2cm+k+1, which is not part of L because the number of c's can be greater than the number of b's.

vxy overlaps a and b in case four.

We would have am+k1bm+k2+2cm+1 if we took uv2xy2z, but that doesn't belong to L because the number of a's can be more than the number of c's.

Case 5: C and B are overlapped by vxy

We would have ambm+2-k1cm+1-k2 if we took uv0xy0z, but that doesn't belong to L because the number of b's and c's can be smaller than the number of a's.

Because of this contradiction and the language's context,

2. Assume that there is a regular use of contradiction in the language. Since it is infinite, the pumping lemma can be used. The pumping lemma integer is denoted by m. We choose a string W where |w|=m and W belongs to L.

Where w=xyz, |xy|=m, and |y|>=1, W = ambmbmam. All xyiz where i>=0 must belong to L if xy is within the first as. Because the expression xy2z=am+kbmbmam does not correspond to the letter L, the language is irregular.

3. As we know context free languages can perform only 1 comparison between two alphabets as if it has one stack with finite automata we can express as an intersection of two context free languages

L is a context free language as we have to compares as with bs and then compare cs with ds

So we will as on stack and pop one a for every b and similarly add c onto the stack and pop one c for every d

L2is also CFG as we will stare all the as onto the stack and then pop b for every c and similarly pop d for every a

So L can be expressed as intersection of two context free languages

**Exercise # 5**

1. Write a regular expression for all strings of 0’s and 1’s in which the total number of zeros to the right of each 1 is even.

b) Write a regular expression for all strings of 0’s and 1’s in which at least one copy of the substring 01 occurs before any copy of the substring 10 occurs in the string. If there is no occurrence of the substring 10 then there need not be any copy of the substring 01.

c) Write a regular expression for all strings of 0’s and 1’s in which there is an even number of 0’s between any two 1’s.

Answer:

1. 100(00)\*(100(00)\*)\*
2. (0(1U0)\*U(0\*U1\*))
3. 1(00(00)\*1)\*

Exercise # 6

A two-way pushdown automaton may move on its input tape in two directions.

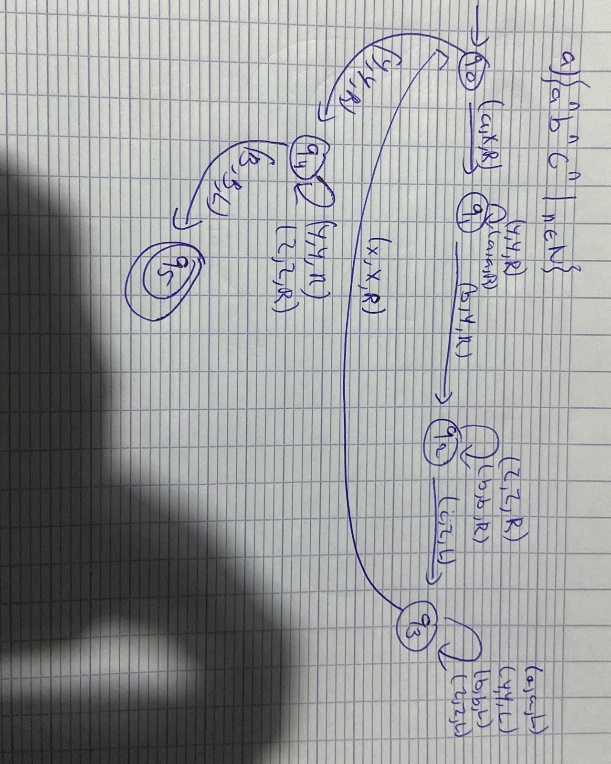
As usual for two-way automata we assume that the begin and end of the input tape is marked by special symbols. In this way the automaton can recognize those positions.

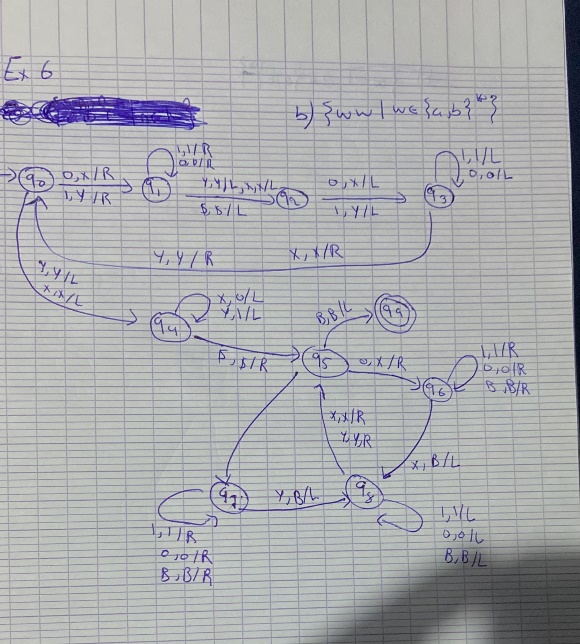
Describe a two-way pda for each of the following languages.

(a) { an bn cn | n ∈ N } (easy)

(b) { ww | w ∈ {a, b}∗ }

(c) { v c uvw | u, v, w ∈ {a, b}∗ }



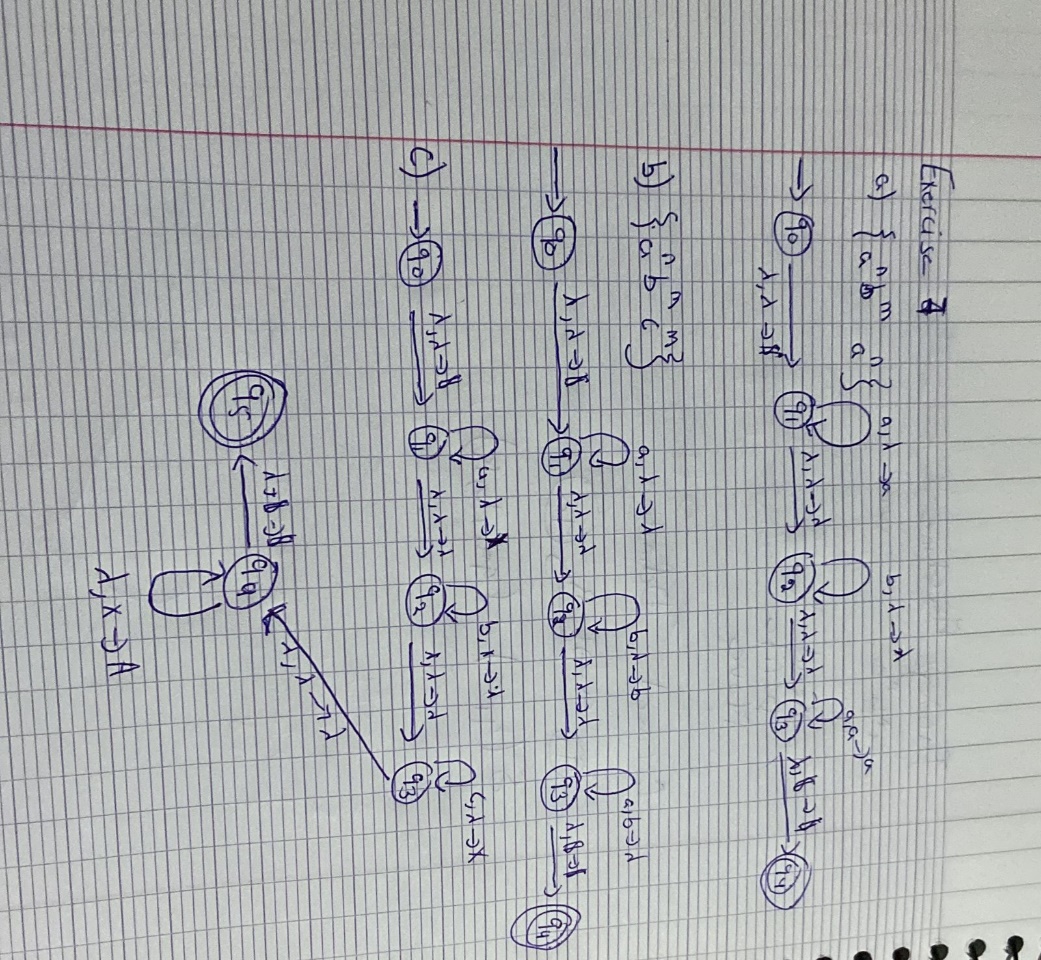


Exercise # 7

Construct pushdown automata for the following languages. Acceptance either by empty stack or by final state.

(a) { an bm an | m, n ∈ N }

(b) { an bm cm | m, n ∈ N }

(c) {ai bj ck | i, j, k ∈ N, i > j}

Exercise # 8

In each case, draw a TM that computes the indicated function. In the first five parts, the function is from N to N. In each of these parts, assume that the TM uses unary notation-that is, the natural number n is represented by the string 1n.

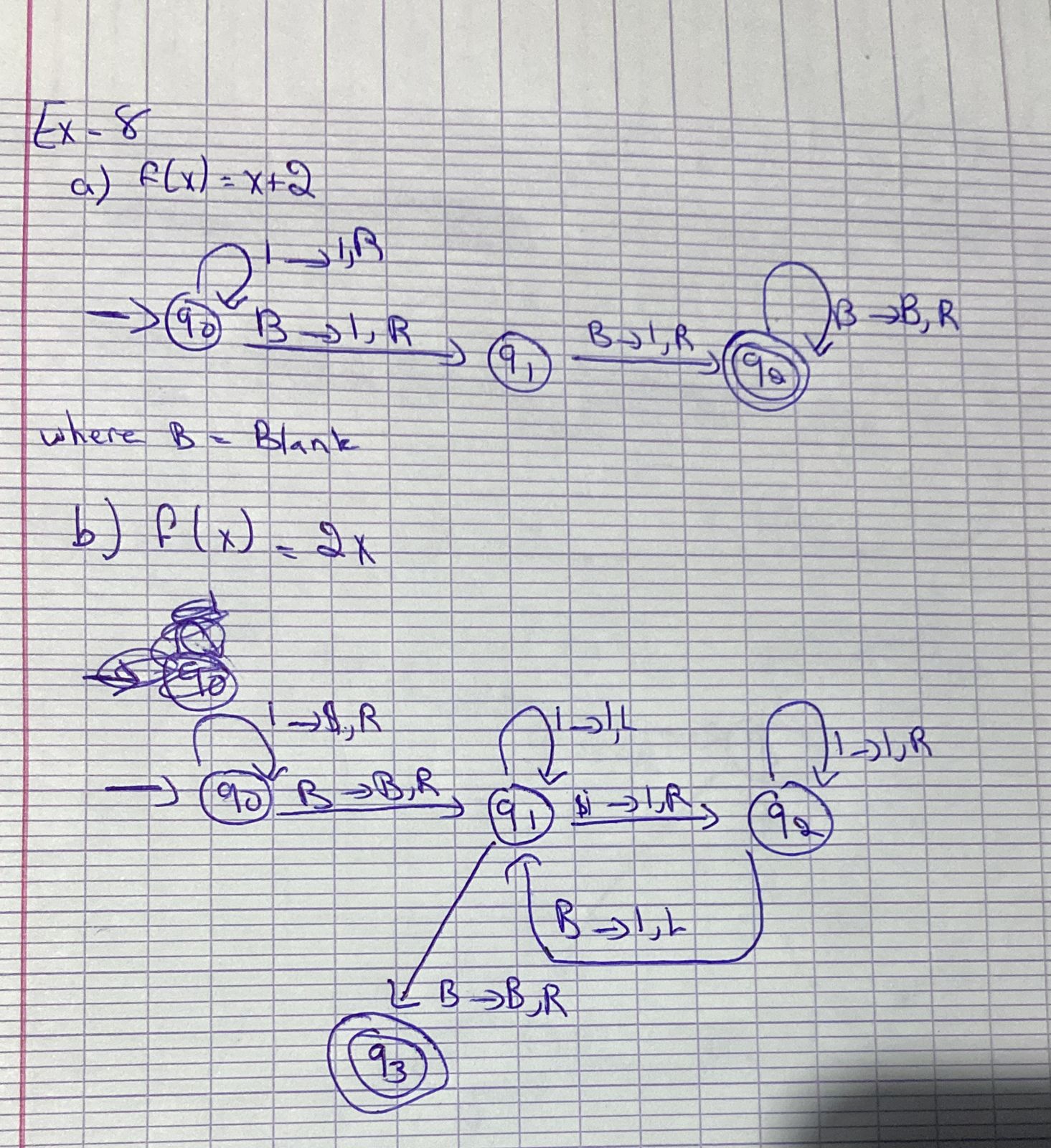
a- f(x)=x+2

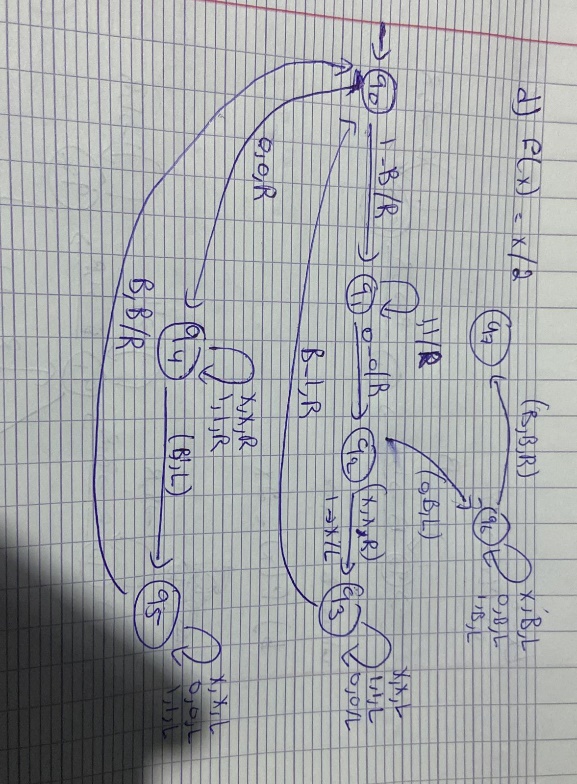
b- f(x)=2x

c- f(x)=x2

d- f(x)=x/2 (“/” means integer division)

Answers:





Exercise #9

Find context-free grammars for the following languages.

Explain your answers.

(a) L = { w : w starts and ends with the same symbol, and w {a, b}\* }

(b) The complement of the language L = {an, bn}

(c) L = { w : na(w) = 2nb(w); where w {a, b}\*}

Answer:

a)

S→ aAa | bAb

A→ bA|aA| λ

b)

S🡪aA|Aa|Bb|bB

A🡪aB|Ba| λ

B🡪bA|Ab| λ

c)

S→ aaSb|bSaa| abSa| λ

**Exercise # 10**

Rewrite each of the following regular expressions as simpler expressions describing the same language:

a. \* | a\* | b\* | (a|b)\*

b. ((a\* b\*)\* (b\*a\*)\*)\*

c. (a\*b)\* | (b\*a)\*

d. (a|b)\* a (a|b)\*

Answer:

a) ∅\* = {ε}, and ε ⊆ (a ∪ b)\*.

a\* ⊆ (a ∪ b)\*.

b\* ⊆ (a ∪ b)\*. So since the first three terms describe subsets of the last one, unioning them into the last

one doesn't add any elements. Thus we can write simply (a ∪ b)\*.

b) To solve this one, we'll use some identities for regular expressions. We don't have time for an extensive

study of such identities, but these are useful ones:

((a\*b\*)\* (b\*a\*)\*)\* =

Using (A\*B\*)\* = (A ∪ B)\* (Both simply describe any string that is composed of elements of

A and elements of B concatenated together in any order)

((a ∪ b)\*(b ∪ a)\*)\* =

Using (A ∪ B) = (B ∪ A) (Set union is commutative)

((a ∪ b)\*(a ∪ b)\*)\* =

Using A\*A\* = A\*

((a ∪ b)\*)\* =

Using (A\*)\* = A\*

(a ∪ b)\*

c) (a\*b)\* ∪ (b\*a)\* = (a ∪ b)\* (In other words, all strings over {a, b}.) How do we know that? (a\*b)\* is the

union of ε and all strings that end in b. (b\*a)\* is the union of ε and all strings that end in a. Clearly any string

over {a, b} must either be empty or it must end in a or b. So we've got them all.

1. (aUb)\*a(aUb)\*

**Exercise # 11**

Indicate whether each of the following is true or false:

e. baa  L(a\*b\*a\*b\*)

f. L(b\*a\*)  L(a\*b\*) = L(a\*)  L(b\*)

g. L(a\*b\*)  L(b\*c\*) = 

h. abcd  L(a(cd)\*b)\*

Answer:

e) True. Consider the defining regular expression: a\*b\*a\*b\*. To get baa, take no a's, then one b, then two

a's then no b's.

f) True. We can prove that two sets X and Y are equal by showing that any string in X must also be in Y

and vice versa. First we show that any string in b\*a\* ∩ a\*b\* (which we'll call X) must also be in a\* ∪ b\*

(which we'll call Y). Any string in X must have two properties: (from b\*a\*): all b's come before all a's; and

(from a\*b\*): all a's come before all b's. The only way to have both of these properties simultaneously is to be

composed of only a's or only b's. That's exactly what it takes to be in Y.

Next we must show that every string in Y is in X. Every string in Y is either of the form a\* or b\*. All strings

of the form a\* are in X since we simply take b\* to be b0

, which gives us a\* ∩ a\* = a\*. Similarly for all strings

of the form b\*, where we take a\* to be a0

.

g) False. Remember that to show that any statements is false it is sufficient to find a single counterexample:

ε ∈ a\*b\* and ε ∈c\*d\*. Thus ε ∈ a\*b\* ∩ c\*d\* , which is therefore not equal to ∅.

h) False. There is no way to generate abcd from (a(cd)\*b)\*. Let's call the language generated by

(a(cd)\*b)\* L. Notice that every string in L has the property that every instance of (cd)\* is immediately

preceded by a. abcd does not possess that property.



a. What is the final configuration if the input is #ab#?

b. What is the final configuration if the input is #baa#?

c. Describe what the Turing machine does for an arbitrary input string in {a,b}\*.

Answer:

a) #ab#

State Input string Action

Q0 #ab# (q1,#,R)

Q1 #ab# (q1,a,R)

Q1 #ab# (q1,b,R)

Q1 #ab# (q2,#,L)

Q2 #ab# (q5,#,R)

Q5 #a## (q6,b,R)

Q6 #a#b# (q7,b,L)

Q7 #a#bb (q7,b,L)

Q7 #a#bb (q2,#,L)

Q2 #a#bb (q3,#,R)

Q3 ###bb (q4,a,R)

Q4 ##abb (q4,b,R)

Q4 ##abb (q4,b,R)

Q4 ##abb# (q7,a,L)

Q7 ##abba (q7,b,L)

Q7 ##abba (q7,b,L)

Q7 ##abba (q7,a,L)

Q7 ##abba (q2,#,L)

Q2 ##abba (q2,#,Y)

Final configuration is abba

b) #baa#

State Input string Action

Q0 #baa# Q1,#,r

Q1 #baa# Q1,b,r

Q1 #baa# Q1,a,r

Q1 #baa# Q1,a,r

Q1 #baa# Q2,#,L

Q2 #baa# Q3,#,r

Q3 #ba## Q4,a,r

Q4 #ba#a# Q7,a,l

Q7 #ba#aa Q7,a,l

Q7 #ba#aa Q2,#,l

Q2 #ba#aa Q3,#,r

Q3 #b##aa Q4,a,r

Q4 #b#aaa Q4,a,r

Q4 #b#aaa Q4,a,r

Q4 #b#aaa# Q7,a,l

Q7 #b#aaaa Q7,a,l

Q7 #b#aaaa Q7,a,l

Q7 #b#aaaa Q7,a,l

Q7 #b#aaaa Q2,#,l

Q2 #b#aaaa Q5,#,r

Q5 ###aaaa Q6,b,r

Q6 ##baaaa Q6,a,r

Q6 ##baaaa Q6,a,r

Q6 ##baaaa Q6,a,r

Q6 ##baaaa Q6,a,r

Q6 ##baaaa# Q7,b,l

Q7 ##baaaab Q7,a,l

Q7 ##baaaab Q7,a,l

Q7 ##baaaab Q7,a,l

Q7 ##baaaab Q7,a,l

Q7 ##baaaab Q2,#,l

Q2 ##baaaab Q2,#,Y

Final configuration is baaaab

c) The turing machine gives the wwR of a certain string W

**Exercise # 13**

True or false:

(a) The context free languages are closed under union.

(b) The context free languages are closed under intersection

(c) The context free languages are closed under Kleene star.

(d) The context free languages are closed under complementation

(e) The context free languages are closed under concatenation

(f) All regular languages are context-free.

(g) A language L is context-free if there is a push-down automaton M such that L = L(M).

Answer:

1. True
2. False
3. True
4. False
5. True
6. True but not all context-free languages are regular
7. True

**Exercise # 14**

Consider the context free grammar G = (V,,R,S) where V is {S,A,B,a,b,c},  is {a,b,c} and R consists of the following rules:

S 🡪 A A 🡪 aS A 🡪 a

S 🡪 B B 🡪 bS B 🡪 b

Is this grammar ambiguous? Justify your answer.

Answer:

S 🡺A🡺aS🡺aB🡺abS🡺abB🡺abb

S🡺A🡺aS🡺aB🡺abS🡺abA🡺aba

S🡺B🡺bS🡺bA🡺baS🡺baA🡺baa

So by that if we draw 2 trees it will have different tree with different strings as we see in the derivation each time we have a different string.

**Exercise # 15**

Consider the push-down automaton M = (K,,,,s,F) where K = {s,f},  = {a,b},  = {a,b}, F = {f}, and  consists of the following transitions:

((s,a,e),(s,a)), ((s,b,e),(s,b)), ((s,a,e),(f,e)), ((f,a,a),(f,e)), ((f,b,b),(f,e)).

(a) Which of the following words are in L(M)? Circle all that are.

abab, aaa, aab, bbbaaabbb, aabbaa, aaaabb

(b) Describe the language accepted by M in simpler terms (that is, without reference to a push-down automaton).

Answer:

