Student Information

Full Name : Koray Can YURTSEVEN
Id Number : 2099547

Answer 1

a.

Table 1: Rules after removing e transition

		Rule of G
S->	aSXaX	R1
S->	aSaX	R2
S->	aSXa	R3
S->	aSa	R4
S->	bSXbX	R5
S->	bSbX	R6
S->	bSXb	R7
S->	bSb	R8
S->	c	R9
X->	aX	R10
X->	a	R11
X->	bX	R12
X->	b	R13

Note that, I have removed e transitions for simplicity.

Table 2: Transitions

Transition Left Side	Transition Right Side	Transition Name
(p,a,e)	(p,a)	T0
(p,e,XaXSa)	(p,S)	T1
(p,e,XaSa)	(p,S)	T2
(p,e,aXSa)	(p,S)	T3
(p,e,aSa)	(p,S)	T4
(p,e,XbXSb)	(p,S)	T5
(p,e,XbSb)	(p,S)	T6
(p,e,bXSb)	(p,S)	T7
(p,e,bSb)	(p,S)	T8
(p,e,c)	(p,S)	T9
(p,e,Xa)	(p,X)	T10
(p,e,a)	(p,X)	T11
(p,e,Xb)	(p,X)	T12
(p,e,b)	(p,X)	T13
(p,e,S)	(q,e)	T14

b.

Table 3: Solution for w = abbcbabbaa

Step	State	Unread	Stack	Transition Used	Rule of G
0	p	abbcbabbaa	e	-	-
1	p	bbcbabbaa	a	T0	-
2	p	bcbabbaa	ba	T0	-
3	p	cbabbaa	bba	T0	-
4	p	babbaa	cbba	T0	-
5	p	babbaa	Sbba	T9	R9
6	p	abbaa	bSbba	T0	-
7	p	abbaa	Sba	T8	R8
8	p	bbaa	aSba	T0	-
9	p	baa	baSba	T0	-
10	p	baa	XaSba	T13	R13
11	p	baa	XSba	T10	R10
12	p	aa	bXSba	T0	-
13	p	aa	Sa	T7	R7
14	p	a	aSa	T0	-
15	p	a	XSa	T11	R11
16	p	e	aXSa	T0	-
17	p	e	S	T3	R3
18	q	e	e	T14	-

Answer 2

a.

Table 4: Solution of TM for q2.a

Current state and symbol read	Next state and symbol changed	Description
qi,U	qo,->	Initial, move right to read first character
q0,1	q1,->	If we read 1, change state and move head to right
q1,1	q0,->	Oscilate between q0 and q1 till we face blank character
q1,U	qh,1	When we encounter with blank character, if we are in state q1, that means the length of the string is odd. Place the current position to 1, and finish reading
q0,U	q2,<-	String length is even, Change state and go left
q2,1	q2,<-	Go left till find blank at the beginning
q2,U	q3,->	We've reached the beginning. Go one right
q3,1	q3,0	Change 1 to 0, stay at the current position
q3,0	q4,->	If we see 0 in q3, change state q4 and find the end of the input
q4,1	q4,->	See above
q4,U	q5,<-	When we reached the right end of the input, go one left and make your state q5
q5,1	q5,U	If we are in q5, make the current char. blank. In this way we are deleting half of 1's at each iteration
q5,U	q6,<-	Go left till find 0
q6,1	q6,<-	Go left till find 0
q6,0	q7,->	We have found the first 0 from right. Go one right and loop
q7,1	q7,0	Make the current char 0
q7,0	q3,0	Loop
q7,U	q8,<-	When we finished the loop, we are in q7,U transition. Make our state q8 to go at the beginning of the input to change 0's to 1's.
q8,0	q8,<-	Find the beginning of the string
q8,U	q9,->	Make state q9 and go right to change 0's to 1's
q9,0	q9,1	
q9,1	q9,->	
q9,U	qh,U	We have converted all 0's to 1's. We have restored the string in problem definition's format. We are done

Answer 3

Given transition rule $\delta(q, x) = (p, y, d)$ where q is the current state, x is the content of the current cell, p is the new state, y is the replacing symbol and d is the direction, when we move only to the right, d becomes irrelevant. Also, it is unnecessary to overwrite something because we always move to the right. Even if we write over something, that becomes useless. Therefore we can ignore y and d. Hence, we will have $\delta(q, x) = (p)$, i.e. transition for Finite Automata.

Answer 4

a.

A queue machine can be defined as a five tuple:
$M = (K, \Sigma, \delta, s, H)$ where,
K is a finite set of states,
Σ is the finite set of the input alphabet, plus blank character and the starting symbol
s is the initial state, and it is an element of K,
$H \in K$ is the halting state,
δ is a transition function from all of the states except the halting states with reading an input to all of the states with reading an element, plus (up, element),(down, element), (right, element), (left, element), where:
$left$ is front
$right$ is rear
up is enqueue
and the $down$ is dequeue

b.

c.

d.

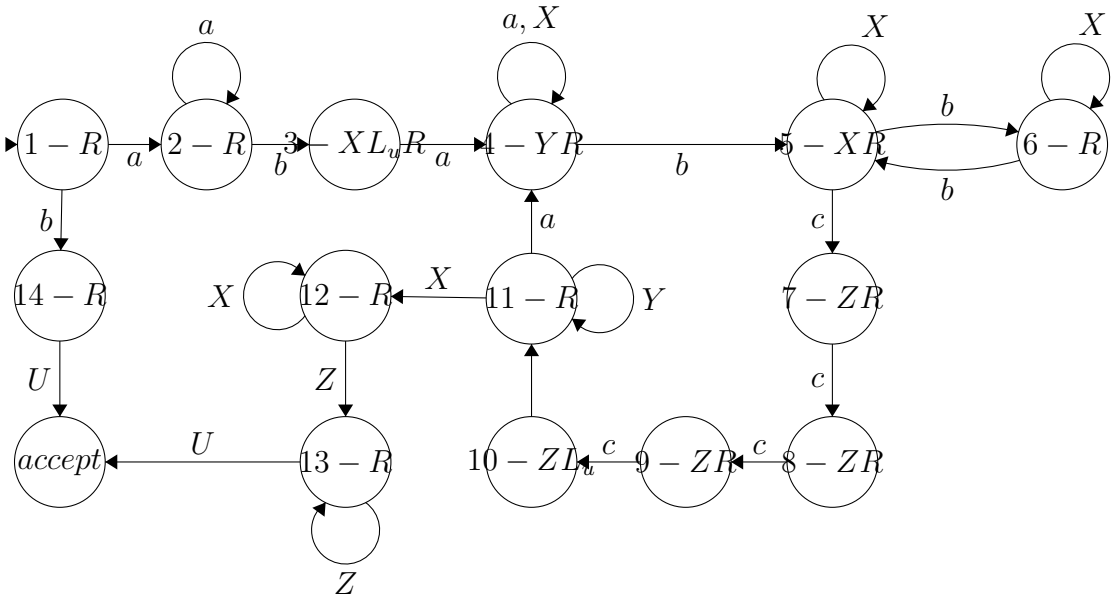
They are equivalent because:
i) Accessing front element in queue TM, go left until reaching the start symbol and after reaching the start symbol, go one element right.
ii) Accesing rear element in queue TM, go right until reaching the blank symbol and after reaching the blank symbol, go one element left.
iii) For enqueuing an element in queue TM, go right until reaching the blank symbol, replace the blank symbol with the element.
iv) For dequeueing an element in queue TM, we can move the start point to one right position or we can shift all the element to the left by one position.
Note that the actions above are in the standard TM.

v For going left and right in the standard TM, just use dequeue and enqueue in queue TM.

e.

Answer 5

a.



Let me explain the notations above. Since I couldn't draw Turing Machine, I draw Finite Automaton and give each circle a number. Normally, there shouldn't be a number and a circle, but these are for clarifying my solutions. It is easy to draw finite automaton.Also, I didn't include the transition that corresponds to the "no" state. All transitions that are not listed above yields to "no" states. If I draw those transitions, my graph will be very complicated.
First, I assume that the given Turing Machine tape will be like in the following: $SUabbcccUU...$ where S denotes start, U denotes blank character, and rest are the string.
When we start at circle number one, we immediately move head to the right and start read the character under the head.
There are 2 options. If we read a b, we move to the circle 14 and move Right. And if read a blank character in here, that means we have reached the end of the input. We can accept this state. This string consists of only b.
Second option is we saw a. Then move in second circle. Whenever we see an a, we move head to the right. When we see a b, we move to the 3rd circle.
We replace this b with X and we do transition L_u to find beginning of the string. When we reached the beginning of the string, we move right, under the head, there will be an a.
When we read an a, we move to the 4th circle, and we are replacing the current character, aka the a with Y. In the 4th circle, whenever I see an a or an X, I move to the right. I'm not replacing read symbols with Y. I'm only looping over the R symbol, which moves head to the right. Whenever I

see a b, I move to the 5th circle.
In the 5th circle, when we are in the first iteration, we have 2 choice. The remaining characters are consist of b's and c's. Also, since we marked the first b with an X, if the given string is correct, we should have $(2^n - 1)$ b's in the string now. i.e. If I have 8 b in the string at first, before I visit 5th circle, I must have 7 b's. In the 5th circle, I change my b to X, then if I see a b, I move to the 6th circle. In 6th circle, I skip X's and when I see a b, I mark it as an X and move to the 5th state.
One thing to mention here is that, when I do this loop in 2nd time, the square loop consists of 4-5-6-7-8-9-10-11-4, I have some X's and some b's in my string. Therefore I need to do a transition between 5 and 6 to skip those X's. Also, every time I run over this loop, I'm marking half of the b's with X. Since I have marked first b in the 3rd circle, I have guaranteed that $(2^n - 1)$ b will end in n cycles.
In the 5th cycle, when I have finish b's, I should have encounter with a c. So for every a, I should have change 3 c's into Z's and move right to read next c.
In the 10th circle, I have finished marking 3 c, so I should go to the beginning of the string.
In the 11th circle, I'm going right to find the first a. If I find an a, I mark that a as a Y, and do the loop 4-5-6-7-8-9-10-11-4. In this loop, for every a, I delete half of the b's, and 3 c.
Again in the 11th circle, if I didn't encounter with an a after I skipped Y's, if the given string is correct, I expect that the character under the head must be a X, aka former b.
So, in circles 12 and 13, if the given string is correct, my string should be consist of as: YYXXXXZZZZZZ. I go right to find an blank character. When I find a blank character, it means that the given string is in the language, and the Turing Machine accepts that string.
The not drawn transitions always lead us to the not accepting state. For example, in the middle of conversion of my b's to X's, if I see an a, given string is not acceptable. It should go to the "no" state and halts. Also, while reading 3 consecutive c's, I'm not expecting any character other than c.
In the 11th circle, if I don't encounter with an a, my string must be consist of Y's, X's and Z's only. If any a, b, c character stayed in those states, that means the string is not in the language. While reading in 12 and 13, if I encounter with any of those characters, the machine must halt and reject the given string.

b.

Answer 6

a.

Since L_1, L_2, L_3 are regular or context free languages, they can be represented by a machine.

In the forth point, if there is a TM that decides the given language, then there is also a TM that decides complement of L_4. Because it is decidable it is a recursive language. Also its complement L_4 (complement of complement of L_4) is also a recursive language. Therefore we can find a machine that accepts this language.

In the fifth point, since it is generated by a grammar, it is a recursively enumerable language. We can find a machine that accepts this language.

M_1, M_2, M_3, M_4, M_5 exist.

b.

Yes, since they are accepted by Turing Machines, we can find an algorithm for membership problems associated with them.

c.

d.

No, not always. Recursively enumerable languages (semi-decidable) are not closed under complement.
RE promises us an acceptor, not a decider. Therefore, if we flip the machine, we'll have a machine that says "no" for the word is not in the language but loop otherwise. That is not a RE.
Also, proof by contradiction:
1- If L and its complement are both semi-decidable, then both are decidable.
2- Hence, if RE is closed under complement, we'll have R=RE.
3- But, it must R $\subset$ RE (by halting problem, it is semi-decidable), so it is a contradiction.