Submission number:	89920
Submission certificate:	FI379023
Submission time:	2014-05-19 17:27:40 PST (GMT - 8:00)

Number of questions:	20
Positive points per question:	4.0
Negative points per question:	0.0
Your score:	80

1. For the grammar:

```
S → AB | CD
A → BC | a
B → AC | C
C → AB | CD
D → AC | d
```

1. Find the generating symbols. Recall, a grammar symbol is *generating* if there is a deriviation of at least one terminal string, starting with that symbol.
2. Eliminate all *useless productions* --- those that contain at least one symbol that is not a generating symbol.
3. In the resulting grammar, eliminate all symbols that are not *reachable* --- they appear in no string derived from S.

In the list below, you will find several statements about which symbols are generating, which are reachable, and which productions are useless. Select the one that is FALSE.

a) A → *a* is useless.

b) C → AB is useless.

c) D is not reachable.

d) C is not reachable.

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

All terminals are generating. The nonterminals A and D evidently derive terminal strings *a* and *d*, respectively. However, B and C are not generating. To see why, notice that each production for B and C has either B or C in the body. Then, since B and C are not generating, and each production for S has a B or C in the body, S is not generating.

Consequently, the only productions that are useful (not useless) are A → a and D → d. Therefore, the only symbol reachable from S is S itself.

2. A Turing machine *M* with start state q_0 and accepting state q_f has the following transition function:

δ(q,a)	**0**	**1**	**B**
q_0	$(q_0,1,R)$	$(q_1,1,R)$	(q_f,B,R)
q_1	$(q_2,0,L)$	$(q_2,1,L)$	(q_2,B,L)
q_2	-	$(q_0,0,R)$	-
q_f	-	-	-

Deduce what *M* does on any input of 0's and 1's. Hint: consider what happens when *M* is started in state q_0 at the left end of a sequence of any number of 0's (including zero of them) and a 1. Demonstrate your understanding by identifying the true transition of *M* from the list below.

a) $q_0$0101 |-* 1010Bq_f

b) $q_0$0101 |-* 1110Bq_f

c) $q_0$0101 |-* 1010q_f

d) $q_0$0011 |-* 1101Bq_f

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

M inverts all 0's and 1's on its input and then accepts. To see why, notice that for any string *w*, *M* makes the following sequence of transitions:

(q_0,0...01w) |-* 1...1$q_0$1w |- 1...11q_1w |- 1...1$q_2$1w |- 1...10q_0w

Also, started in state q_0 with only 0's to its right, *M* moves to the right, replacing the 0's by 1's, and accepts when it reaches a blank.

3. Consider the languge L={a}. Which grammar defines L?

a) G_1:S → AB|C|a, A → b, C → a

b) G_1:S → ac|a, A → c|b|ε

c) G_1:S → AC|a|c, A → c|ε

d) G_1:S → AC|ε, A → b

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

There are a few points worth remembering:

1. Many of the grammars have useless productions --- productions where one or more of the nonterminals have no productions. We can delete all useless productions without changing the language.
2. If a grammar generates any string other than *a*, it is a wrong answer, even if it generates *a*.

4. Consider the grammars:

G_1:$S \rightarrow AB \mid a \mid abC$, $A \rightarrow b$, $C \rightarrow abC \mid c$

G_2:$S \rightarrow a \mid b \mid cC$, $C \rightarrow cC \mid c$

These grammars do not define the same language. To prove, we use a string that is generated by one but not by the other grammar. Which of the following strings can be used for this proof?

a) abababcc

b) ababababcc

c) cccc

d) abac

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

Grammar G_1 does not use the first production for S because there is no production for B. It generates C to be $(ab)^i c$, i=0,1,.... Then it uses the last production for S to generate all strings $(ab)^i c$, i=1,2,... Finally it uses the second production of S to generate string a.

Thus, G_1 produces the language $\{a, (ab)^i c, i=1,2,...\}$.

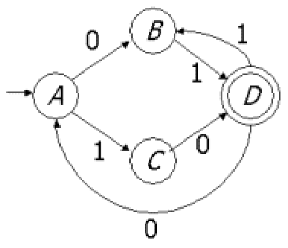
Grammar G_2 produces language $\{a,b,c^i, i=2,3,...\}$

Explanation of correct choices:

Strings b,ccccccc,cc,ccc,cccc: not generated by G_1.

Strings abc, ababc, abababc, ababababababc, abababababc: not generated by G_2.

5. Here is a finite automaton:



Which of the following regular expressions defines the same language as the finite automaton? Hint: each of the correct choices uses component expressions. Some of these components are:

1. The ways to get from A to D without going through D.
2. The ways to get from D to itself, without going through D.
3. The ways to get from A to itself, without going through A.

It helps to write down these expressions first, and then look for an expression that defines all the paths from A to D.

a) (01+10)(11+(01+10)0)*

b) (01+10)(11+0(01+10))*

c) (01+10)(0(01+10))*(11)*

d) (01+10)(11*+0(01+10))*

Answer submitted: **b)**

You have answered the question correctly.

Question Explanation:

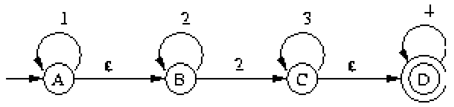
Let us start with an expression for the paths that go from A to D without passing through D. One can only go through B or C, so this expression is AD=(01+10).

Next, how do we get from D to itself without passing through D? We can either go to B and then back to D, on input 11, or we can go on a 0 to A, and then to D as in the paragraph above. The expression for this case is 0(01+10), and the expression for all ways to go from D to D without passing through D is DD=11+0(01+10).

To get from A to A without passing through A, we must go to D and follow the arc from D to A, labeled 0. Since we can stay in the BD loop as long as we like, the expression for ways to go from A to A without passing through A is AA=(01+10)(11)*0.

Now, two reasonable ways to describe all paths from A to D are AD(DD)*=(01+10)(11+0(01+10))* and (AA)*AD(11)*=((01+10)(11)*0)*(01+10)(11)*. The first expression is justified by the idea that all ways to get from A to D get to D for the first time, and then go from D to D zero or more times. The second expression is just ified by the idea that we can go from A to A as many times as we like, but eventually we leave A for the last time, and arrive at D. We can then go from D to D several times, without returning to A; the latter can only be done by input sequence 11.

6. Here is a nondeterministic finite automaton with epsilon-transitions:



Suppose we use the extended subset construction from Section 2.5.5 (p. 77) to convert this epsilon-NFA to a deterministic finite automaton with a dead state, with all transitions defined, and with no state that is inaccessible from the start state. Which of the following would be a transition of the DFA?

Note: we use S-x->T to say that the DFA has a transition on input x from state S to state T.

a) {C,D}-4->{D}

b) {A,B,C}-3->{C,D}

c) {B}-2->{B,C}

d) {C,D}-ε->{D}

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

Here is the transition table of the DFA, with an arrow indicating the start state and *'s indicating accepting states.

	1	**2**	**3**	**4**
→{A,B}	{A,B}	{B,C,D}	{}	{}
* {B,C,D}	{}	{B,C,D}	{C,D}	{D}
* {C,D}	{}	{}	{C,D}	{D}
* {D}	{}	{}	{}	{D}

{}	{}	{}	{}	{}

7. Here are eight simple grammars, each of which generates an infinite language of strings. These strings tend to look like alternating *a*'s and *b*'s, although there are some exceptions, and not all grammars generate all such strings.

1. S → abS | ab
2. S → SS | ab
3. S → aB; B → bS | a
4. S → aB; B → bS | b
5. S → aB; B → bS | ab
6. S → aB | b; B → bS
7. S → aB | a; B → bS
8. S → aB | ab; B → bS

The initial symbol is S in all cases. Determine the language of each of these grammars. Then, find, in the list below, the pair of grammars that define the same language.

a) G1: S → aB, B → bS, B → b
   G2: S → aB, B → bS, S → b

b) G1: S → aB, B → bS, B → a
   G2: S → aB, B → bS, B → b

c) G1: S → abS, S → ab
   G2: S → SS, S → ab

d) G1: S → aB, B → bS, B → ab
   G2: S → SS, S → ab

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

There are four grammars found among the correct answers:

1. S → abS | ab
2. S → SS | ab
3. S → aB; B → bS | b
4. S → aB | ab; B → bS

All generate the language described by regular expression $(ab)^+$, that is, one or more repetitions of ab.

(1) is evidently an unambiguous way of defining this language. (2) is an ambiguous grammar, but also evidently defines the same language.

For (3), notice that each time we use S → aB we must replace the B immediately by either b or bS. Thus, each S is replaced by either ab or abS, and grammar (3) behaves essentially like grammar (1), with extra steps to replace the B's.

For (4), B can only be replaced by bS, so the use of S → aB, together with the use of the lone production for B has the same effect as S → abS. Thus, grammar (4) is also essentially the same as (1).

8. Apply the CYK algorithm to the input ababaa and the grammar:

```
S → AB | BC
A → BA | a
```

```
B → CC | b
C → AB | a
```

Compute the table of entries X_{ij} = the set of nonterminals that derive positions i through j, inclusive, of the string ababaa. Then, identify a true assertion about one of the X_{ij}'s in the list below.

a) X_{15} = {B}

b) X_{26} = {S,A}

c) X_{34} = {C}

d) X_{16} = {S,A,C}

Answer submitted: **b)**

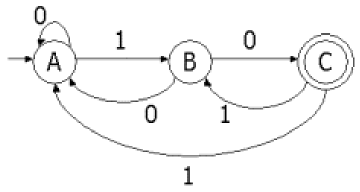
You have answered the question correctly.

Question Explanation:

Here is the table:

B					
SAC	SA				
B	B	SA			
B	SC	B	-		
SC	SA	SC	SA	B	
AC	B	AC	B	AC	AC
a	**b**	**a**	**b**	**a**	**a**

9. Convert the following nondeterministic finite automaton:



to a DFA, including the dead state, if necessary. Which of the following sets of NFA states is **not** a state of the DFA that is accessible from the start state of the DFA?

a) {A}

b) {A,C}

c) {C}

d) {}

Answer submitted: **c)**

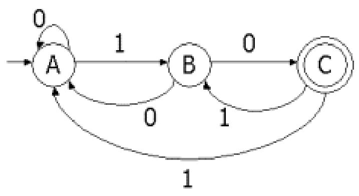
You have answered the question correctly.

Question Explanation:

Here is the state transition table of the DFA constructed using the subset construction, but using only states that are reachable from the start state {A}.

	0	**1**
{A}	{A}	{B}
{B}	{A,C}	{}
{A,C}	{A}	{A,B}
{}	{}	{}
{A,B}	{A,C}	{B}

10. When we convert an automaton to a regular expression, we need to build expressions for the labels along paths from one state to another state that do not go through certain other states. Below is a nondeterministic finite automaton with three states. For each of the six orders of the three states, find regular expressions that give the set of labels along all paths from the first state to the second state that never go through the third state.



Then identify one of these expressions from the list of choices below.

a) 0*1 represents the paths from A to B that do not go through C.

b) 1 represents the paths from C to A that do not go through B.

c) 1(0+1010)* represents the paths from C to A that do not go through B.

d) 0(0+10)* represents the paths from B to A that do not go through C.

Answer submitted: **d)**

You have answered the question correctly.

Question Explanation:

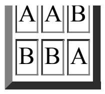
Here is a table of the expressions:

From	To	Without Passing Through	Expression
A	B	C	(0+10)*1
A	C	B	φ
B	A	C	0(0+10)*
B	C	A	(01)*0
C	A	B	10*
C	B	A	(10)*1

For example, consider the first row: going from A to B without passing through C. The last step is surely the arc labeled 1 from A to B. However, we can stay in A using 0 or oscillate between A and B using 10 as many times as we wish before that final step. The paths that take us from A to A without going through C are expressed by (0+10)*. Thus, the paths from A to B that never touch C can be expressed (0+10)*1.

11. Here is the transition function of a simple, deterministic automaton with start state A and accepting state B:

	0	**1**



A	A	B
B	B	A

We want to show that this automaton accepts exactly those strings with an odd number of 1's, or more formally:

δ(A,w) = B if and only if w has an odd number of 1's.

Here, δ is the extended transition function of the automaton; that is, δ(A,w) is the state that the automaton is in after processing input string w. The proof of the statement above is an induction on the length of w. Below, we give the proof with reasons missing. You must give a reason for each step, and then demonstrate your understanding of the proof by classifying your reasons into the following three categories:

A)
Use of the inductive hypothesis.

B)
Reasoning about properties of deterministic finite automata, e.g., that if string s = yz, then δ(q,s) = δ(δ(q,y),z).

C)
Reasoning about properties of binary strings (strings of 0's and 1's), e.g., that every string is longer than any of its proper substrings.

Basis (|w| = 0):

(1)
w = ε because ________

(2)
δ(A,ε) = A because ________

(3)
ε has an even number of 0's because ________

Induction (|w| = n > 0)

(4)
There are two cases: (a) when w = x1 and (b) when w = x0 because ________
Case (a):

(5)
In case (a), w has an odd number of 1's if and only if x has an even number of 1's because ________

(6)
In case (a), δ(A,x) = A if and only if w has an odd number of 1's because ________

(7)
In case (a), δ(A,w) = B if and only if w has an odd number of 1's because ________
Case (b):

(8)
In case (b), w has an odd number of 1's if and only if x has an odd number of 1's because ________

(9)
In case (b), δ(A,x) = B if and only if w has an odd number of 1's because ________

(10)
In case (b), δ(A,w) = B if and only if w has an odd number of 1's because ________

a) (8) for reason B.

b) (9) for reason B.

c) (8) for reason C.

d) (9) for reason C.

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

Here is the proof with reasons filled in:

Basis ($|w| = 0$):

(1) $w = \varepsilon$ because ε is the only string of length 0 (C).

(2) $\delta(A,\varepsilon) = A$ because an automaton goes nowhere on input ε (B).

(3) ε has an even number of 0's because no zeroes is an even number of zeroes (C).

Induction ($|w| = n > 0$)

(4) There are two cases: (a) when w = x1 and (b) when w = x0 because a binary string of length n can only end in 0 or 1 (C).

Case (a):

(5) In case (a), w has an odd number of 1's if and only if x has an even number of 1's because x has one fewer 1 than does w (C).

(6) In case (a), $\delta(A,x) = A$ if and only if w has an odd number of 1's because the inductive hypothesis applies to x, which is of length n-1 (A).

(7) In case (a), $\delta(A,w) = B$ if and only if w has an odd number of 1's because there are transitions from A to B and from B to A on input 1 (B).

Case (b):

(8) In case (b), w has an odd number of 1's if and only if x has an odd number of 1's because x and w have the same number of 1's (C).

(9) In case (b), $\delta(A,x) = B$ if and only if w has an odd number of 1's because the inductive hypothesis applies to x, which is of length n-1 (A).

(10) In case (b), $\delta(A,w) = B$ if and only if w has an odd number of 1's because there are transitions from A to A and from B to B on input 0 (B).

12. If we convert the context-free grammar G:

```
S → AS | A
A → 0A | 1B | 1
B → 0B | 0
```

to a pushdown automaton that accepts L(G) by empty stack, using the construction of Section 6.3.1, which of the following would be a rule of the PDA?

a) $\delta(q,\varepsilon,S) = \{(q,AS), (q,A)\}$

b) $\delta(q,1,A) = \{(q,B), (q,\varepsilon)\}$

c) $\delta(q,\varepsilon,S) = \{(q,AS)\}$

d) $\delta(q,\varepsilon,A) = \{(q,0A)\}$

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

There is one state, q. The input symbols are 0 and 1, and the stack symbols are {S, A, B, 0, 1}. S is the initial stack symbol. The rules are:

$\delta(q,\varepsilon,S) = \{(q,AS), (q,A)\}$
$\delta(q,\varepsilon,A) = \{(q,0A), (q,1B), (q,1)\}$
$\delta(q,\varepsilon,B) = \{(q,0B), (q,0)\}$
$\delta(q,0,0) = \{(q,\varepsilon)\}$
$\delta(q,1,1) = \{(q,\varepsilon)\}$

13. Identify in the list below a sentence of length 6 that is generated by the grammar $S \rightarrow (S)S \mid \varepsilon$

a)) (( ) ( )

b) (( ))) (

c) (( )) ( )

d))) ((( )

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

This grammar generate all strings of balanced parentheses and only strings of balanced parentheses. Each left parenthesis in a string generated by this grammar is matched by a unique following right parenthesis. Likewise, each right parenthesis in a sring generated by this grammar is matched by a unique preceding left parenthesis. See Sections 5.1.3 (p. 175) and 5.1.5 (p. 179) for definitions of derivations and languages.

14. We wish to perform the reduction of acceptance by a Turing machine to MPCP, as described in Section 9.4.3 (p. 407). We assume the TM *M* satisfies Theorem 8.12 (p. 346): it never moves left from its initial position and never writes a blank. We know the following:

1. The start state of *M* is *q*.
2. *r* is the accepting state of *M*.
3. The tape symbols of *M* are 0, 1, and B (blank).
4. One of the moves of *M* is $\delta(q,0) = (p,1,L)$.

Which of the following is DEFINITELY NOT one of the pairs in the MPCP instance that we construct for the TM *M* and the input 001?

a) (#, #q001#)

b) (#, #)

c) (r##, #)

d) (#, #q001)

Answer submitted: **d)**

You have answered the question correctly.

Question Explanation:

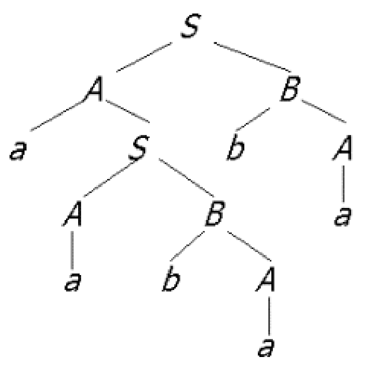
We know we need the following pairs:

1. (#, #q001#) --- the starting pair (rule 1).
2. (0,0), (1,1), and (#,#) by rule (2).

3. (0q0, p01) and (1q0, p11) by rule (3).
4. All pairs by rule (4) of the form (xry, r), (xr, r), and (ry, r), where *x* and *y* are each either 0 or 1.
5. (r##, #) by rule (5)

[illegible]

15. The folowing is a parse tree in some unknown grammar G:



Which of the following productions is **definitely not** a production of G?

a) None of the other choices.

b) A → a

c) S → aC

d) B → b

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

If all you see is a parse tree of G, you can tell some productions that *must* be in G, but you can never tell for certain that there isn't some production of G that happens not to have been used in the parse tree you see. Thus, the only correct answer is "none of the other choices is correct."

16. Here are seven regular expressions:

1. (0*+10*)*
2. (0+10)*
3. (0*+10)*
4. (0*+1*)*
5. (0+1)*
6. (0+1*0)*
7. (0+1*)*

Determine the language of each of these expressions. Then, find in the list below a pair of equivalent expressions.

a) (0+1)* and (0*+1*)*

b) (0+1*0)* and (0+1*)*

c) (0+10)* and (0+1*)*

d) (0+1)* and (0+10)*

Answer submitted: **a)**

You have answered the question correctly.

Question Explanation:

Expressions 1, 4, 5, and 7 all generate all strings of 0's and 1's. They are thus all equivalent. Expressions 2 and 3 are equivalent; each generates all strings of 0's and 1's such that every 1 is followed by a 0. Expression 6 is equivalent to none of the others, since it generates all strings of 0's and 1's that are either empty or end in 0.

17. Let *h* be the homomorphism defined by h(a) = 01, h(b) = 10, h(c) = 0, and h(d) = 1. If we take any string *w* in (0+1)*, h^{-1}(w) contains some number of strings, N(w). For example, h^{-1}(1100) = {ddcc, dbc}, i.e., N(1100) = 2. We can calculate the number of strings in h^{-1}(w) by a recursion on the length of *w*. For example, if w = 00x for some string x, then N(w) = N(0x), since the first 0 in w can only be produced from *c*, not from *a*.

Complete the reasoning necessary to compute N(w) for any string *w* in (0+1)*. Then, choose the correct value of N(1011010).

a) 9

b) 64

c) 15

d) 21

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

First, notice that N(ε) = 1; i.e., h^{-1}(ε) = {ε}. Also, N(0) = N(1) = 1, since h^{-1}(0) = {c} and h^{-1}(1) = {d}.

For the inductive step, If w = 00x or w = 11x, then the first symbol of *w* comes from *c* or *d*, respectively, while 0x or 1x can come from N(0x) or N(1x) different strings. Thus, N(00x) = N(0x) and N(11x) = N(1x).

Now, consider the case when w = 01x. Then *w* comes from either a string ay where h(y) = x, or *w* comes from a string cy, where h(y) = 1x. Thus, N(01x) = N(1x) + N(x). If w = 10x, the reasoning is analogous; N(10x) = N(0x) + N(x).

We now have a way to count h^{-1}(w) for any *w*. For example, if w = 00110, we reason:

- N(ε) = 1 (basis).
- N(0) = 1 (basis).
- N(10) = N(0) + N(ε) = 2 (induction, with x = ε).
- N(110) = N(10) = 2.
- N(0110) = N(110) + N(10) = 4.
- N(00110) = N(0110) = 4.

18. Suppose one transition rule of some PDA P is δ(q,0,X) = {(p,YZ), (r,XY)}. If we convert PDA P to an equivalent context-free grammar G in the manner described in Section 6.3.2 (p. 247), which of the following could be a production of G derived from this transition rule? You may assume *s* and *t* are states of P, as well as *p*, *q*, and *r*.

a) [qXs] → 0[pYt][pZs]

b) [qXs] → 0[rXt][tYs]

c) [qXs] → 0[qXt][tYr]

d) [qXs] → [rXt][tYs]

Answer submitted: **b)**

You have answered the question correctly.

Question Explanation:

If *m* and *n* are any states of P, then the fact that (p,YZ) is in δ(q,0,X) says that there will be a production [qXm] → 0[pYn][nZm]. Similarly, the choice (r,XY) says that [qXm] → 0[rXn][nYm] is a production.

19. Here is the transition table of a DFA:

	0	**1**
→A	E	D
*B	A	C
C	G	B
D	E	A
*E	H	C
F	C	B
G	F	E
H	B	H

Find the minimum-state DFA equivalent to the above. Then, identify in the list below the pair of equivalent states (states that get merged in the minimization process).

a) D and G

b) D and H

c) A and G

d) A and B

Answer submitted: **b)**

You have answered the question correctly.

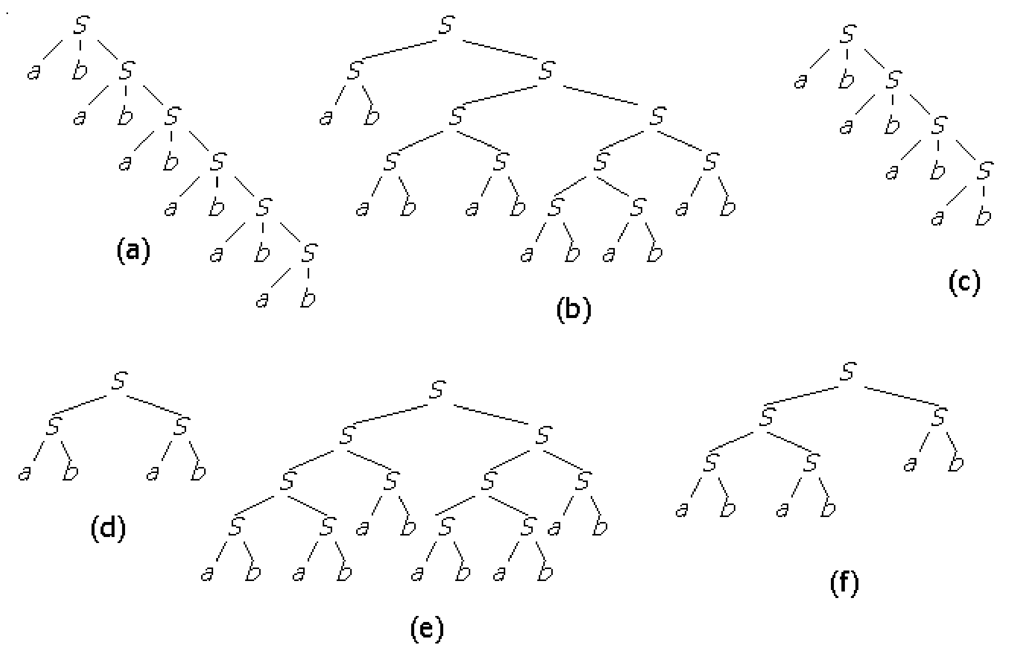
Question Explanation:

Here is the table of distinguishabilities:

A							
B	x	x					
C	x	x	x				
D			x	x			
E	x	x		x	x		
F	x	x	x		x	x	
G	x	x	x		x	x	
	H	**A**	**B**	**C**	**D**	**E**	**F**

We start by noting that accepting states B and E are distinguishable from all the other states. In this problem, each of the other x's in the table can be filled in immediately because the pair of distinguishable states goes, on either 0 or 1, to one state that is accepting and another that is not.

20. Consider the grammar G: S → abS, S → ab. Which of the following strings is a word of L(G) AND is the yield of one of the parse trees for grammar G in the figure below?



a) abababababab

b) ababS

c) abababab

d) ababababab

Answer submitted: **c)**

You have answered the question correctly.

Question Explanation:

L(G) consists of all strings of one or more repetitions of ab. However, only trees (a) and (c) match the grammar G, and the yields of these trees are abababababab and abababab, respectively.