Submission number: 88458
Submission certificate: EI669316
Submission time: 2014-05-18 16:04:12 PST (GMT - 8:00)

Number of questions: 25
Positive points per question: 4.0
Negative points per question: 0.0
Your score: 92

1. Consider the grammar G and the language L:

G: S → AB | a | abC, A → b, C → abC | c

L: {w | w a string of a's, b's, and c's with an equal number of a's and b's}.

Grammar G does not define language L. To prove, we use a string that either is produced by G and not contained in L or is contained in L but is not produced by G. Which string can be used to prove it?

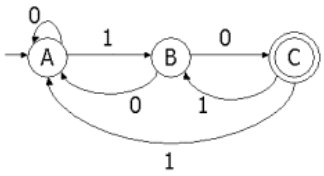
a) aba
b) cccabab
c) abac
d) abacccc

Answer submitted: **a)**

Your answer is incorrect.

This string does not belong in L and also is not generated by G. See Section 5.1.3 (p. 175) for a discussion of how strings are generated by a grammar.

2. 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}
b) {}
c) {B}
d) {B,C}

Answer submitted: **d)**

You have answered the question correctly.

3. The intersection of two CFL's need not be a CFL. Identify in the list below a pair of CFL's such that their intersection is not a CFL.

a) $L_1=\{aba^nb^nc^i \mid n>0, i>0\}$
$L_2=\{aba^nb^jc^i \mid n>0, i>0, j>0\}$

b) $L_1=\{aba^ib^nc^iba \mid n>0, i>0\}$
$L_2=\{aba^nb^ic^iba \mid n>0, i>0, j>0\}$

c) $L_1=\{aba^nb^nc^iba \mid n>0, i>0\}$
$L_2=\{aba^nb^ic^iba \mid n>0, i>0\}$

d) $L_1=\{aba^nb^nc^nba \mid n>0, i>0\}$
$L_2=\{aba^nb^ic^iba \mid n>0, i>0, j>0\}$

Answer submitted: **c)**

You have answered the question correctly.

4. Consider the grammar G1: $S \rightarrow \varepsilon$, $S \rightarrow aS$, $S \rightarrow aSbS$ and the language L that contains exactly those strings of a's and b's such that every prefix has at least as many a's as b's. We want to prove the claim: G1 generates all strings in L.

We take the following inductive hypothesis to prove the claim:

For $n < k$, G1 generates every string of length n in L.

To prove the inductive step we argue as follows:

"For each string w in L either _____(a1) or _____(a2) holds. In both cases we use the inductive hypothesis and one of the rules to show that string w can be generated by the grammar. In the first case we use rule $S \rightarrow aS$ and in the second case we use rule $S \rightarrow aSbS$."

Which phrases can replace the _____ so that this argument is correct?

a) a1: w can be written as w=aw'bw" where for both w' and w" it holds that each prefix has as many a's as b's.
a2: each prefix has more a's than b's.

b) a1: each prefix has more a's than b's.
a2: there is a unique b in string w such that for the part of the string until the b (b also included) each prefix has as many a's as b's and for the part after b each prefix has as many a's as b's.

c) a1: each prefix has equal number of b's and a's.
a2: w can be written as w=aw'bw" where for both w' and w" it holds that each prefix has as many a's as b's.

d) a1: w can be written as w=aw' where for each prefix of w' has as many a's as b's.
a2: there is a b in string w such that the part of the string until the b (b also included) has all prefixes with as many a's as b's and the part after this b has all prefixes with as many a's as b's.

Answer submitted: **d)**

You have answered the question correctly.

5. Convert the grammar:

```
S → A | B | 2
A → C0 | D
B → C1 | E
C → D | E | 3
D → E0 | S
E → D1 | S
```

to an equivalent grammar with no unit productions, using the construction of Section 7.1.4 (p. 268). Then, choose one of the productions of the new grammar from the list below.

a) $A \rightarrow C1$

b) $S \rightarrow E$

c) $E \rightarrow E01$

d) $S \rightarrow E1$

Answer submitted: **a)**

You have answered the question correctly.

6. Programming languages are often described using an extended form of context-free grammar, where curly brackets are used to denote a construct that can repeat 0, 1, 2, or any number of times. For example, $A \rightarrow B\{C\}D$ says that an *A* can be replaced by a *B* and a *D*, with any number of *C*'s (including 0) between them. This notation does not allow us to describe anything but context-free languages, since an extended production can always be replaced by several conventional productions.

Suppose a grammar has the extended production:

$A \rightarrow a\{BC\}b$

Convert this extended production to conventional productions. Identify, from the list below, the conventional productions that are equivalent to the extended production above.

a) $A \rightarrow aA_1b$
$A_1 \rightarrow BC \mid \varepsilon$

b) $A \rightarrow aA_1b$
$A_1 \rightarrow A_1BC \mid \varepsilon$

c) $A \rightarrow aA_1b$
$A_1 \rightarrow BCA_1 \mid BC$

d) $A \rightarrow aBCA_1b$
$A_1 \rightarrow BCA_1 \mid \varepsilon$

Answer submitted: **b)**

You have answered the question correctly.

7. A nondeterministic 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_1,0,R)}	{(q_1,0,R)}	{(q_1,0,R)}
q_1	{(q_1,1,R), (q_2,0,L)}	{(q_1,1,R), (q_2,1,L)}	{(q_1,1,R), (q_2,B,L)}
q_2	{(q_f,0,R)}	{(q_2,1,L)}	{}
q_f	{}	{}	{}

Simulate all sequences of 5 moves, starting from initial ID $q_0$1010. Find, in the list below, one of the ID's reachable from the initial ID in EXACTLY 5 moves.

a) $01111q_1$

b) $01101q_1$

c) $01q_210$

d) $011q_211$

Answer submitted: **a)**

You have answered the question correctly.

8. Which of the following strings is NOT in the Kleene closure of the language {011, 10, 110}?

a) 11001110

b) 0111010

c) 01110111

d) 011011110

Answer submitted: **c)**

You have answered the question correctly.

9. 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 δ(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) (r1, r)

b) (0q0, p01)

c) (0, 0)

d) (#, #q001)

Answer submitted: **d)**

You have answered the question correctly.

10. The operation Perm(w), applied to a string w, is all strings that can be constructed by permuting the symbols of w in any order. For example, if w = 101, then Perm(w) is all strings with two 1's and one 0, i.e., Perm(w) = {101, 110, 011}. If L is a regular language, then Perm(L) is the union of Perm(w) taken over all w in L. For example, if L is the language L(0*1*), then Perm(L) is all strings of 0's and 1's, i.e., L((0+1)*).

If L is regular, Perm(L) is sometimes regular, sometimes context-free but not regular, and sometimes not even context-free. Consider each of the following regular expressions R below, and decide whether Perm(L(R)) is regular, context-free, or neither:

1. (01)*
2. 0*+1*
3. (012)*
4. (01+2)*

a) Perm(L((012)*)) is regular.

b) Perm(L((01+2)*)) is regular.

c) Perm(L(0*+1*)) is context-free but not regular.

d) Perm(L((01)*)) is context-free but not regular.

Answer submitted: **d)**

You have answered the question correctly.

11. Here is a grammar, whose variables and terminals are NOT named using the usual convention. Any of *R* through *Z* could be either a variable or terminal; it is your job to figure out which is which, and which could be the start symbol.

```
R → ST | UV
T → UV | W
V → XY | Z
X → YZ | T
```

We do have an important clue: There are no useless productions in this grammar; that is, each production is used in some derivation of some terminal string from the start symbol. Your job is to figure out which letters definitely represent variables, which definitely represent terminals, which could represent either a terminal or a nonterminal, and which could be the start symbol. Remember that the usual convention, which might imply that all these letters stand for either terminals or variables, does not apply here.

a) *Z* could be a variable or a terminal.

b) *T* could be a variable or a terminal.

c) *S* must be a terminal.
d) *X* must be a terminal.

Answer submitted: **c)**

You have answered the question correctly.

12. Identify in the list below a sentence of length 6 that is generated by the grammar S → (S)S | ε

a) (( )) ( )
b)) (( ) ( )
c)))) (((
d) ( )) (( )

Answer submitted: **a)**

You have answered the question correctly.

13. Design the minimum-state DFA that accepts all and only the strings of 0's and 1's that end in 010. To verify that you have designed the correct automaton, we will ask you to identify the true statement in a list of choices. These choices will involve:

1. The number of *loops* (transitions from a state to itself).
2. The number of transitions into a state (including loops) on input 1.
3. The number of transitions into a state (including loops) on input 0.

Count the number of transitions into each of your states ("in-transitions") on input 1 and also on input 0. Count the number of loops on input 1 and on input 0. Then, find the true statement in the following list.

a) There is one state that has one in-transition on input 0.
b) There is one state that has three in-transitions on input 1.
c) There are two states that have two in-transitions on input 0.
d) There is one state that has no in-transitions on input 0.

Answer submitted: **a)**

You have answered the question correctly.

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

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

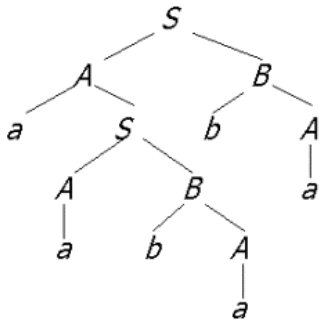
to a pushdown automaton that accepts L(G) by empty stack, using the construction of Section 6.3.1, which of the following would be a rule of the PDA?

a) δ(q,ε,B) = { (q,0)}
b) δ(q,ε,S) = {(q,A)}
c) δ(q,0,0) = {(q,ε)}
d) δ(q,ε,S) = {(q,SA), (q,A)}

Answer submitted: **c)**

You have answered the question correctly.

15. The parse tree below represents a leftmost derivation according to the grammar `S → AB, A → aS|a, B → bA`.



Which of the following is a left-sentential form in this derivation?

a) aABB
b) aabAba
c) aaBbA
d) aAbaba

Answer submitted: **a)**

You have answered the question correctly.

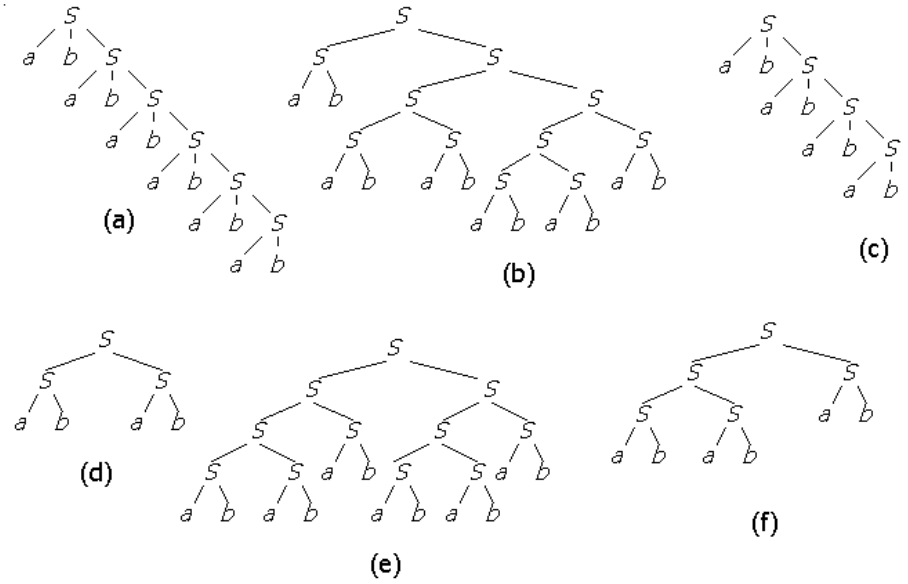
16. h is a homomorphism from the alphabet {a,b,c} to {0,1}. If h(a) = 01, h(b) = 0, and h(c) = 10, which of the following strings is in $h^{-1}(010010)$?

a) *abab*
b) *bcba*
c) *abcb*
d) *baba*

Answer submitted: **a)**

You have answered the question correctly.

17. Which of the parse trees below yield the same word?



a) a and c
b) a and b
c) d and f
d) a and f

Answer submitted: **a)**

Your answer is incorrect.

One yields the word abababababab and the other the word abababab. See Section 5.2.2 (p. 185) on yields of parse trees.

18. Let L be the language of all strings of a's and b's such that no prefix (proper or not) has more b's than a's. Let G be the grammar with productions

```
S → aS | aSbS | ε
```

To prove that L = L(G), we need to show two things:

1. If S =>* w, then w is in L.
2. If w is in L, then S =>* w.

We shall consider only the proof of (2) here. The proof is an induction on n, the length of w. Here is an outline of the proof, with reasons omitted. You need to supply the reasons.

Basis:

1) If n=0, then w is ε because _____.

2) S =>* w because _____.

Induction:

3) Either (a) w can be written as w=aw' where for w' each prefix has as many a's as b's or (b) w can be written as w=aw'bw" where for both w' and w" hold that each prefix has as many a's as b's because _____.

4a) In case (a), w' is in the language because _____.

5a) In case (a), S =>* w' because _____.

6a) In case (a), S =>* w because _____.

4b) In case (b), both w' and w" are in the language because _____.

5b)
In case (b), S =>* w' because _____.
6b)
In case (b), S =>* w" because _____.
7b)
In case (b), S =>* w because _____.

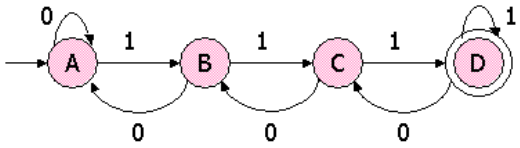
For which of the steps above is the appropriate reason "by the inductive hypothesis"?

a) 3
b) 4b
c) 6b
d) 2

Answer submitted: **c)**

You have answered the question correctly.

19. Examine the following DFA:



This DFA accepts a certain language L. In this problem we shall consider certain other languages that are defined by their tails, that is, languages of the form (0+1)*w, for some particular string *w* of 0's and 1's. Call this language L(w). Depending on *w*, L(w) may be contained in L, disjoint from L, or neither contained nor disjoint from L (i.e., some strings of the form *xw* are in L and others are not).

Your problem is to find a way to classify *w* into one of these three cases. Then, use your knowledge to classify the following languages:

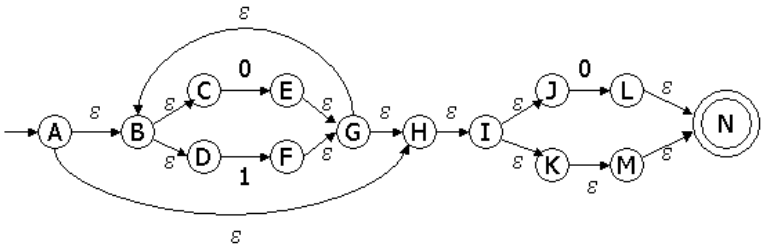
1. L(1111001), i.e., the language of regular expression (0+1)*1111001.
2. L(11011), i.e., the language of regular expression (0+1)*11011.
3. L(110101), i.e., the language of regular expression (0+1)*110101.
4. L(00011101), i.e., the language of regular expression (0+1)*00011101.

a) L(11011) is neither disjoint from L nor contained in L.
b) L(00011101) is neither disjoint from L nor contained in L.
c) L(00011101) is disjoint from L.
d) L(1111001) is disjoint from L.

Answer submitted: **d)**

You have answered the question correctly.

20. Here is an epsilon-NFA:



Suppose we construct an equivalent DFA by the construction of Section 2.5.5 (p. 77). That is, start with the epsilon-closure of the start state A. For each set of states *S* we construct (which becomes one state of the DFA), look at the transitions from this set of states on input symbol 0. See where those transitions lead, and take the union of the epsilon-closures of all the states reached on 0. This set of states becomes a state of the DFA. Do the same for the transitions out of *S* on input 1. When we have found all the sets of epsilon-NFA states that are constructed in this way, we have the DFA and its transitions.

Carry out this construction of a DFA, and identify one of the states of this DFA (as a subset of the epsilon-NFA's states) from the list below.

a) BCDFGHIJK
b) ABCDEFGHIJKLMN
c) LN
d) BCDFGHIJKMN

Answer submitted: **d)**

You have answered the question correctly.

21. What is the concatenation of `abc` and `cda`?

a) `acbdca`
b) `abccda`
c) `abc cda`
d) `abc.cda`

Answer submitted: **b)**

You have answered the question correctly.

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

a) G_1:S → d|a, A → c|b|ε
b) G_1:S → ac|a, A → c|b|ε
c) G_1:S → AB|a, A → b
d) G_1:S → AC|a|ab, A → c|ε

Answer submitted: **c)**

You have answered the question correctly.

23. The binary string 0011011 is a member of which of the following problems? Remember, a "problem" is a language whose strings represent the cases of a problem that have the answer "yes." In this question, you should assume that all languages are sets of binary strings interpreted as base-2 integers. The exception is the problem of finding *palindromes*, which are strings that are identical when reversed, like 0110110, regardless of their numerical value.

a) Is the given string a composite number (not a prime)?
b) Is the given string a palindrome?
c) Is the given string a multiple of 3?
d) Is the given string a perfect square?

Answer submitted: **a)**

You have answered the question correctly.

24. In this question you are asked to consider the truth or falsehood of six equivalences for regular expressions. If the equivalence is true, you must also identify the law from which it follows. In each case the statement R = S is conventional shorthand for "L(R) = L(S)." The six proposed equivalences are:

1. 0*1* = 1*0*
2. 01φ = φ
3. ε01 = 01
4. (0* + 1*)0 = 0*0 + 1*0
5. (0*1)0* = 0*(10*)
6. 01+01 = 01

Identify the correct statement from the list below.

Note: we use φ for the empty set, because the correct symbol is not recognized by Internet Explorer.

a) 0*1* = 1*0* follows from the commutative law of concatenation.

b) (0*1)0* = 0*(10*) follows from the associative law for concatenation.

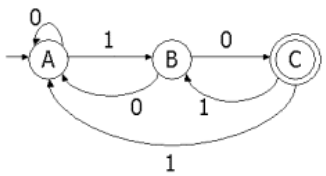
c) (0*1)0* = 0*(10*) is false.

d) 0*1* = 1*0* follows from the associative law for concatenation.

Answer submitted: **b)**

You have answered the question correctly.

25. 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) $(01)^+$ represents the paths from B to C that do not go through A.

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

c) (0+10)*1 represents the paths from A to B that do not go through C.

d) 1+101 represents the paths from C to B that do not go through A.

Answer submitted: **c)**

You have answered the question correctly.