1.	Consider the following grammar: S -> aAab S -> a A c A -> a A -> Of the four language classes we have discussed (regular, context-free, context-sensitive, and unlimited), what is the smallest category into which the language generated by this grammar fits? Justify your answer.	Regular: aaab | aab | aac | ac
2.	Translate the regular expression (a | b)c into an NFA using Thompson's construction. How many states and epsilon edges are there in the resulting NFA?	7 states, 4 epsilon edges
3.	Given the following NFA corresponding to the regular expression a(b|c)*d: When translating this NFA into a DFA using the subset construction algorithm, which NFA states are in the start state of the DFA?	1
4.	in Fortran FORMAT statements there can appear a lexical entity known as a Hollerith constant. It consists of some decimal digits giving a count (say **n**), the letter **H**, and then **n** characters. The overall effect is to form a string constant n characters long. Is it possible to specify the language of Hollerith	No, since a finite automaton cannot compare the number of characters after the H to the count., No, this is a context-sensitive language and a finite automaton cannot recognize a context-sensitive language.

	constants (i.e., the set of only valid Hollerith constants) with an NFA or a DFA?	
5.	Consider the following grammar: S -> a A a b S -> a A c A -> a A -> Which grammar rules are in the LL parse table entry for non-terminal S and column $ (EOF)? I.e., when seeing end of file, how would a parse function parseS() choose to parse the input?	This parse table entry is empty.
6.	Consider the following grammar: S -> a A a b S -> a A c A -> a A -> **Is A nullable?**	Yes
7.	Is it possible to recognize Hollerith constants with JLex using start states? If no, explain why not. If yes, explain how you would do it.	Convert the digits to an int, store it in a local variable, and count down the characters with a start state.
8.	Which of the following languages is a regular language?	{ an b | n >= 0 }, { an bn | 0 <= n <= 10 } and The language of all possible floating-point constants.

9. Consider the following grammar:
S -> a A a b
S -> a A c
A -> a
A ->
Which grammar rules are in the LL parse table entry for non-terminal S and token a? I.e., with lookahead a, how could a parse function parseS() choose to parse the input?

S -> a A a b

S -> a A c

10. Is it possible to write a handwritten scanner that recognizes Hollerith constants? If no, explain why not. If yes, explain how you would do it.

It would be possible to write a handwritten scanner that recognizes Hollerith constants because using the first token as the size we can determine how long the rest will be

11. Consider the following grammar:
S -> a A a b
S -> a A c
A -> a
A ->
** Which terminal symbols are in FOLLOW(S)?
**

$ (indicating end of file)

12. Given ASCII as the underlying character set, the regular expression [a-c] is equivalent to which of the following regular expressions?

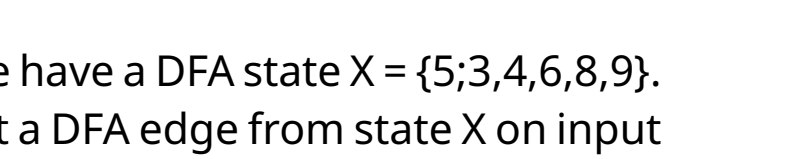
a | b | c

13. (T/F) If for a given grammar, an LR(0) parser has a S/R conflict but an LR(1) parser has no conflict, then the grammar must be SLR as well.

False

14.

	Consider the following grammar: S -> a A a b S -> a A c A -> a A -> ** Is this grammar LL(k)? I.e., can a top-down parser with arbitrary lookahead parse the language according to this grammar? **	Yes, because all sentences in the language are of finite length.
15.	The regular expression ab | c is equivalent to which of the following regular expressions?	(a b) | c
16.	Consider the following grammar: S -> a A a b S -> a A c A -> a A -> ** Which terminal symbols are in FIRST(S)? **	a
17.	The regular expression ab* is equivalent to which of the following regular expressions?	a(b*)
18.	Given the following NFA corresponding to the regular expression a(b|c)*d: ** Suppose we have a DFA state X = {5;3,4,6,8,9}. If we construct a DFA edge from state X on input d, which NFA states are in the resulting DFA state? **	10
19.	** Suppose we limit the length of Hollerith constants to no more than 50 characters. Is it possible to specify the language of Hollerith con-	Yes, because we could construct an automaton where different states correspond to the possible counts

	stants of limited length with an NFA or a DFA? **	and then count down by having a state transition with each character.
20.	Consider the following grammar: S -> a A a b S -> a A c A -> a A -> ** Which terminal symbols are in FIRST(A)? **	a
21.	Consider the following grammar: S -> a A a b S -> a A c A -> a A -> **Which terminal symbols are in FOLLOW(A)?**	a and c
22.	(T/ F) If for a given grammar, an LR(0) parser has a conflict but the conflict can be resolved by an SLR parser, then the grammar is also LALR(1).	True
23.	If for a given grammar, an LALR(1) parser has a conflict then the grammar also cannot be parsed with a recursive-descent parser.	False
24.	An LALR(1) parser can always successfully resolve a S/R conflict by giving preference to shi.	False
25.	Given the following NFA corresponding to the regular expression a(b|c)*d:  ** Suppose we have a DFA state X = {5;3,4,6,8,9}. If we construct a DFA edge from state X on input	3, 4, 5, 6, 8, 9

	b, which NFA states are in the resulting DFA state? **	
26.	Translate the regular expression ab*c into an NFA using Thompson's construction. How many states and epsilon edges are there in the resulting NFA?	6 states 4 epsilon edges
27.	Given the following NFA corresponding to the regular expression a(b|c)*d: ** Suppose we have a DFA state X = {5;3,4,6,8,9}. If we construct a DFA edge from state X on input c, which NFA states are in the resulting DFA state? **	3, 4, 6, 7, 8, 9
28.	Using start states in JLex makes it easier to process certain lexemes (such as translating escape character sequences inside strings), but it is not necessary to use start states. Any language that can be recognized with JLex with start states could also be recognized with JLex without the use of start states.	False
29.	S -> a A a b S -> a A c A -> a A -> Which grammar rules are in the LL parse table entry for non-terminal A and token c? I.e., with lookahead c, how could a parse function parseA() choose to parse the input?	A ->

30. S -> a A a b
S -> a A c
A -> a
A ->

** Which grammar rules are in the LL parse table entry for non-terminal A and token a? I.e., with lookahead a, how could a parse function parseA() choose to parse the input? **

A -> a
A ->
and 1

31. If for a given grammar, an LALR(1) parser has a S/R conflict then the grammar is not LR(1), either.

True

32. Consider the following grammar:
S -> E $
E -> id
E -> id (E)
E -> E + id

** Suppose we are using a shift-reduce parser for parsing either of the inputs "id (id) $" or "id + id$" What parse conflict (if any) does an LR(0) parser have once the first id is on the stack and either (or + would come next in the input? **

** S/R conflict: shift or reduce using rule E->id **

33. Given the following NFA corresponding to the regular expression a(b|c)*d:

** What is the minimum number of states that DFA needs that recognizes the same language as this NFA? You can either translate this NFA into a DFA and then combine equivalent states

3

or construct a DFA by hand. In either case, you should end up with the same DFA. **