CSE431 Midterm Examination

Name: Key

ID Number: _____

- Show your work to receive maximum credit. Partial credit will be given, but only if work is shown.
- Circle or otherwise indicate your answer to each question.
- Please keep your written answers brief and to the point. Incorrect or rambling statements can hurt your score on a question.
- If a question requires you to provide code, please use Java syntax. Minor syntax errors will not be counted against you, but extremely sloppy code will result in deductions.
- This exam is worth 20% of your final grade.
- The exam is broken up into 7 parts and is worth 100 points. There are 35 questions total. It is your responsibility to make sure you have all of the pages.
- You must sign the pledge below for your exam to count. Any cheating will cause the students involved to receive an F for this course. Other action may be taken.

Section	Possible Points	Score
I	10	
II	10	
III	12	
IV	23	
V	12	
VI	33	
Total	100	

Pledge: On my honor, I have neither given nor received any unauthorized aid on this exam.

Signed: _____

1. (10 points) Fill in the blank:

(a) (1 point) A *language* is a set of strings.

(b) (2 points) When an NFA M of n states and t transitions over an alphabet Σ of s symbols is transformed into a DFA D, the number of states in D is at most 2^n.

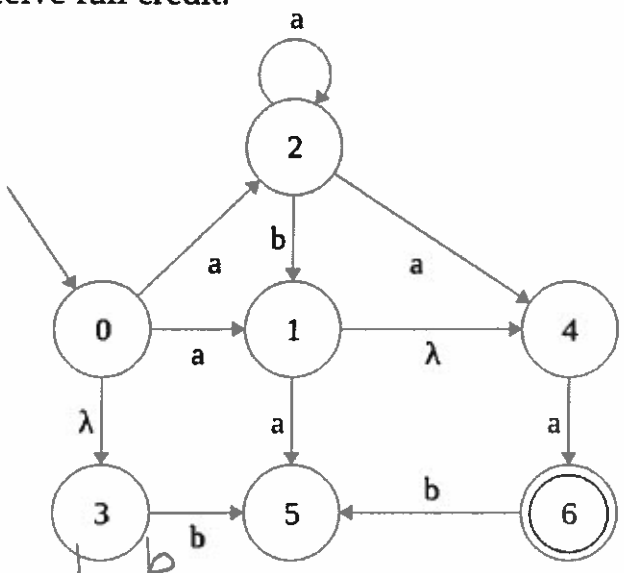
(c) (3 points) Recursive descent parsers operate in O(n) time, with respect to the length of the string.

(d) (1 point) Table-driven top-down parsers are (more / less / equally) efficient when compared to recursive descent parsers.

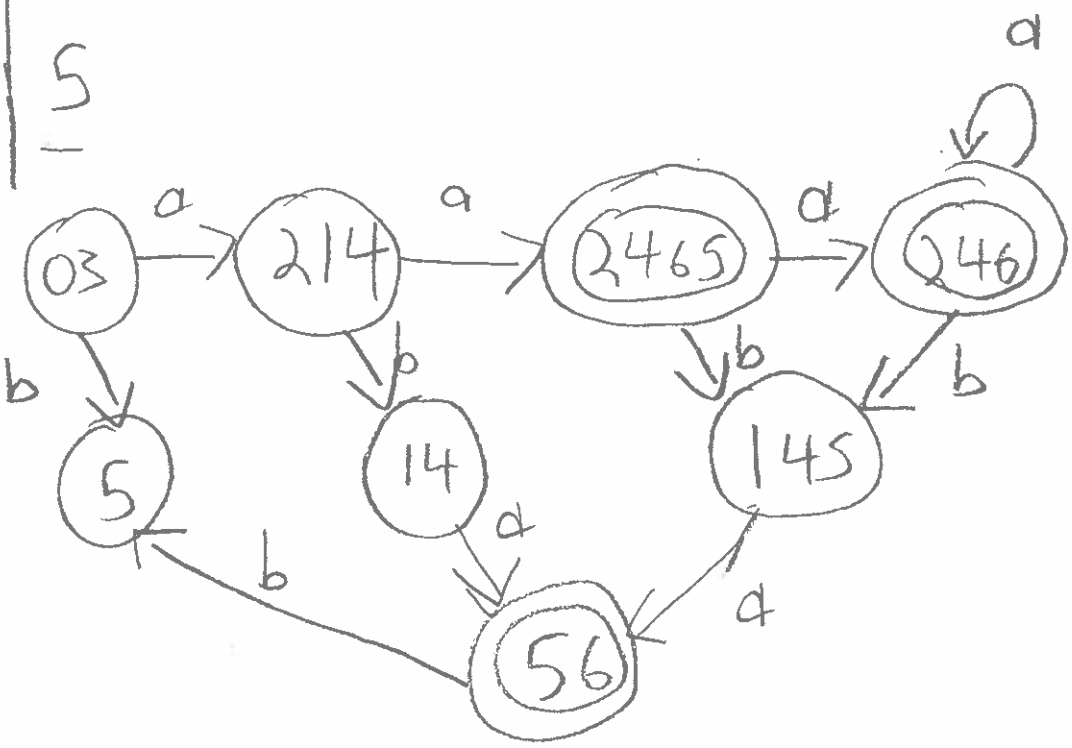
(e) (1 point) True or false: If a grammar is LALR(1), then it is also SLR(1). False.

(f) (2 points) True or false: There is no algorithm that can determine whether an arbitrary context free grammar is ambiguous. True

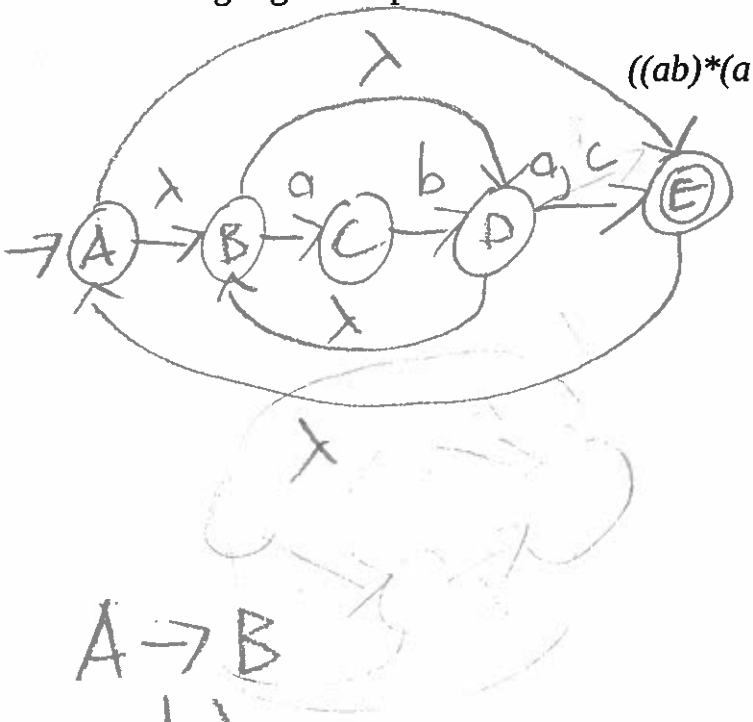
2. (10 points) Eliminate the non-determinism from this NFA. You must show the final DFA to receive full credit.



	a	b
03	214	5
214	2465	14
5	–	–
2465	246	145
246	246	145
145	56	–
56	–	5
14	56	–



3. (12 points) Create a right linear grammar that defines the same language as the following regular expression:



Its easier to construct the NFA then convert it.

A → B
| λ

B → aC
| D

C → bD

D → aA
| cA
| B

4. (23 points) Answer the following:

(a) (15 points) Construct the Derives-λ, FIRST, FOLLOW, and PREDICT sets for the following grammar:

	Predict
S -> AB	a b d
A -> BcD	b d
| aB	a
B -> Db	d b
| bCd	b
C -> cC	c
| d	d
D -> d	d
| λ	b d

	λ	First	Follow
S	F	a b d	—
A	F	a b d	b d d
B	F	b d	c $ d b
C	F	c d	d
D	T	d	b d

(b) (3 points) Is this grammar LL(1)? Explain how you know.

It is not. This grammar is ambiguous since the predict sets of B and D overlap.

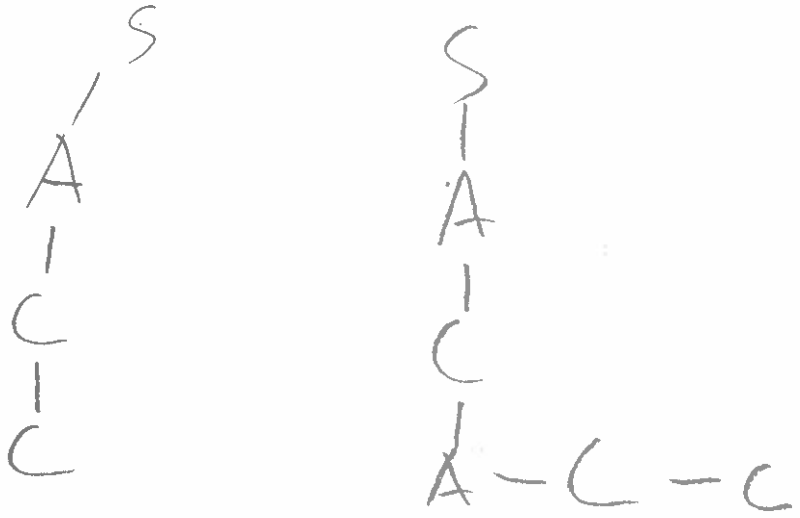
(c) (5 points) Write code that would implement the recursive descent method for the *A* non-terminal of this grammar. No error recovery is needed.

```
public void A() {
  if(peek() == sym.b || peek() == sym.d){
      B();
      expect("c");
      D();
  }else if(peek() == sym.a){
      expect("a");
      B();
  }else{
      oops("Parse error");
  }
}
```

5. (12 points) The following grammar is ambiguous:

S -> A
  | bB
A -> C
  | aB
  | aA
B -> bC
  | b
C -> bB
  | A
  | c

(a) (4 points) Prove the ambiguity of this grammar.



(b) (8 points) Fix the grammar by removing the ambiguity. Use your answer from part a along with your new grammar to show that it is no longer ambiguous.

I will not ask you to do a problem like this.

S→A | bB
A→bB
aB
aA
C

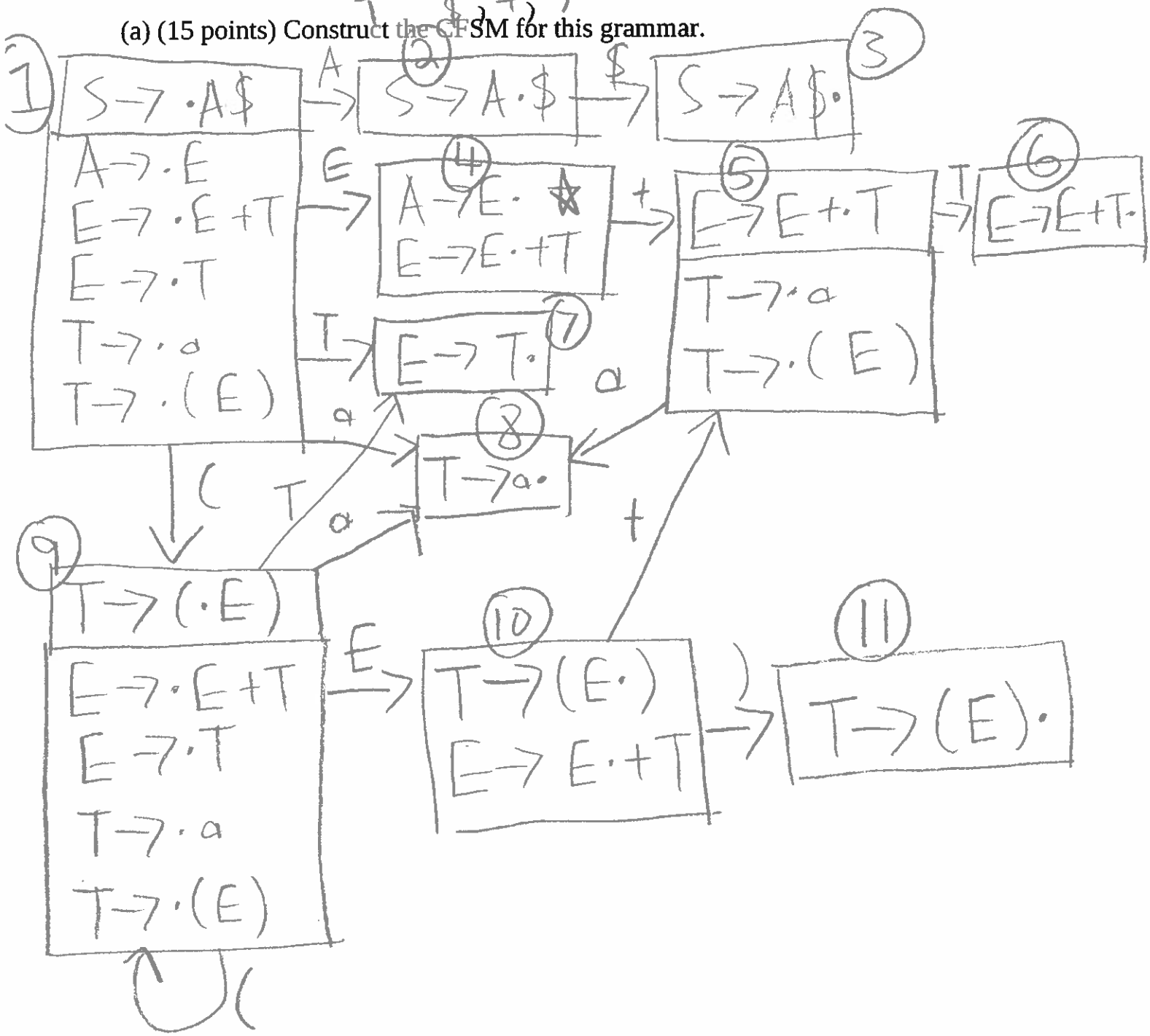
B→bA
| b

6. (33 points) Examine the following grammar:

1 S -> A$
2 A -> E
3 E -> E + T
4 | T
5 T -> a
6 | (E)

	Follow
S	
A	$
E	$, +,)
T	$, +,)

(a) (15 points) Construct the CFSM for this grammar.



(b) (6 points) Answer the following:

(2 points) Is this grammar LR(0)? Why or why not?

No, there is a conflict.

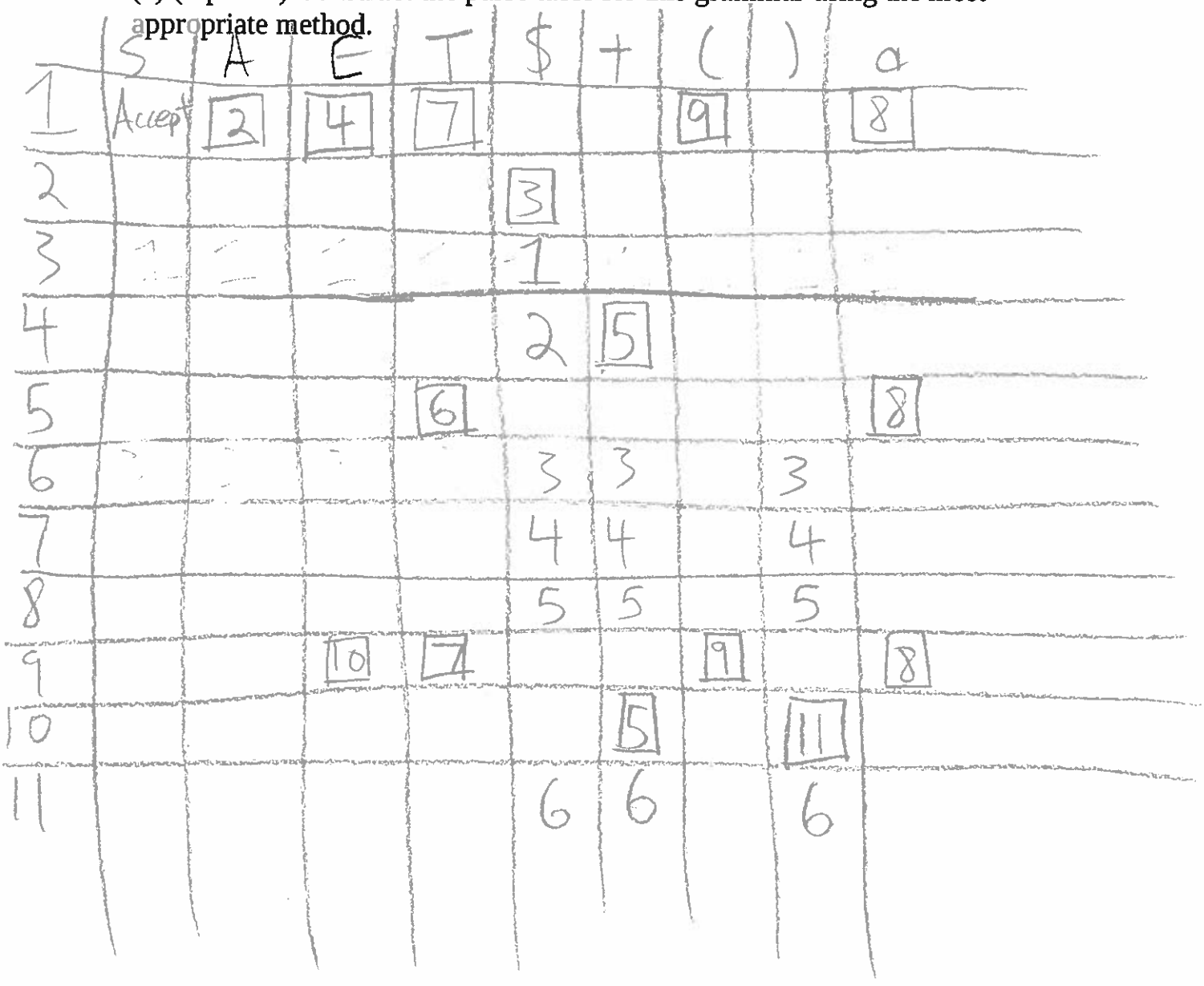
(2 points) Is this grammar SLR(1)? Why or why not?

Yes, the follow set resolves the conflict.

(2 points) Is this grammar LALR(1)? Why or why not?

Yes, because it is SLR(1).

(c) (8 points) Construct the parse table for this grammar using the most appropriate method.



(e) (4 points) Construct the parse tree for the string "(a + a) + (a)". Circle the portion of the parse tree that would represent the first reduction to take place during the bottom up parse process.

