Midterm I - Solution
CS164, Spring 2014

March 3, 2014

- Please read all instructions (including these) carefully.
- **This is a <u>closed-book exam</u>. You are allowed a one-page handwritten cheat sheet.**
- Write your name, login, and SID.
- There are TODO pages in this exam and 3 questions, each with multiple parts. If you get stuck on a question move on and come back to it later.
- You have 1 hour and 15 minutes to work on the exam.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. **Do not** use any additional scratch paper.
- Solutions will be graded on correctness and ***clarity***. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.
- No electronic devices are allowed, including **cell phones** used merely as watches. Silence your cell phones and place them in your bag.

LOGIN: ______________________

NAME: ______________________

SID: ______________________

Problem	Max points	Points
1	28	
2	24	
3	48	
Sub Total	100	

1 Regular Expressions and Finite Automata

Consider a small language using only the letters “z”, “o”, and the slash character “/”. A comment in this language starts with “/o” and ends after the very next “o/”.

(a) Give a regular expression that matches exactly one complete comment and nothing else. Assume comments do not nest. For full credit, use only the core notations: ε, “ab”, AB, A|B, and A^* **[8 points]**

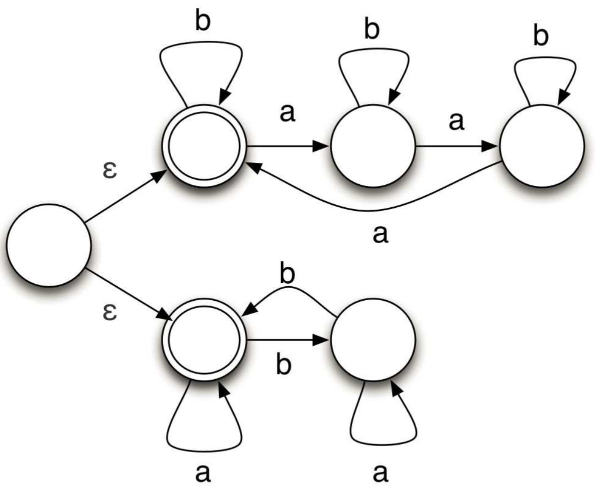
/o((o*z)|/)*o*o/

Consider the language over the alphabet Σ = {a,b} containing strings in which number of 'a's is a multiple of 3 or number of 'b's is a multiple of 2.

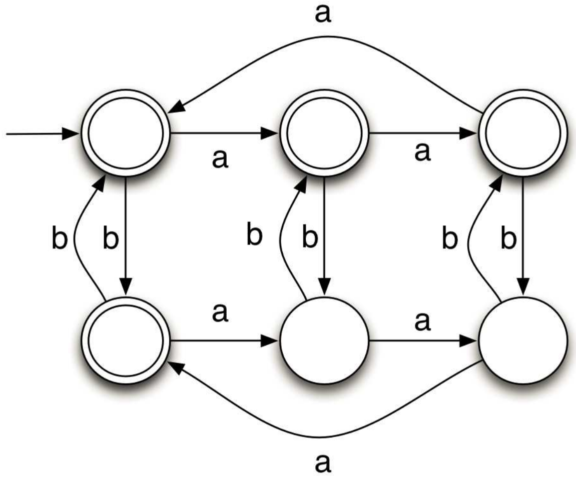
(b) Write a regular expression for this language. **[4 points]**

(a*ba*ba*)* | (b*ab*ab*ab*)*|a*|b*

(c) Write an NFA for this language. **[4 points]**



(d) Write a DFA with at most 6 states for this language **[4 points]**



(3) Can we construct a regular expression for a language over the alphabet Σ = {a,b} whose strings have equal number of occurrences of a and b? Explain. **[4 points]**

No. Corresponding state machine requires infinite number of states.

Yes/No (1)
Reason (3)

2 LL Parsing

Consider the following grammar with terminals *, !, n, (, and).

E → F H
H → * E | ϵ
F → F ! | G
G → n | (E)

(a) The grammar is not LL(1). Explain in one sentence why **[2 points]**

F -> F ! is left recursive

(b) Fix the grammar to make it LL(1) by filling in the blanks below **[4 points]**

E	→	F H	F	→	____ G K________
H	→	* E | ϵ	K	→	__! K! K__ ____
G	→	n | (E)	K	→	____ ϵ ____

(c) Compute the first and the follow set of the fixed grammar **[8 points]**

	First	Follow
E	(, n	$,)
F	(, n	$,), *
G	(, n	$,), *, !
H	*, ϵ	$,)
K	!, ϵ	$,), *

(d) Compute LL(1) parsing table **[10 points]**

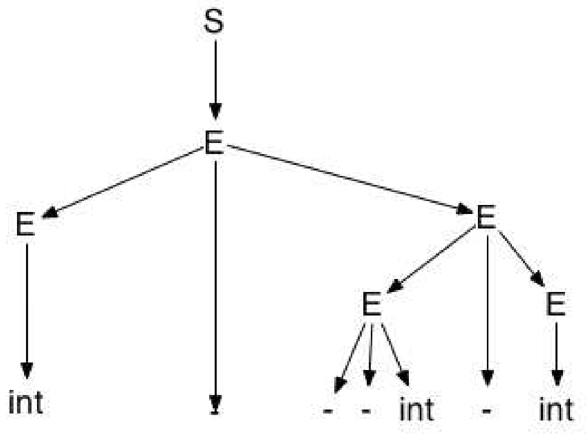
	*	!	(	)	$	n
E			E → FH			E → FH
F			F → GK			F → GK
G			G → (E)			G → n
H	H → *E			ϵ	ϵ	
K	ϵ	K → !K		ϵ	ϵ	

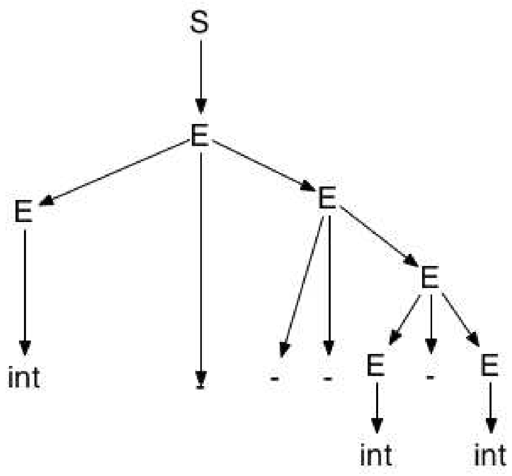
3 LR Parsing and Ambiguity

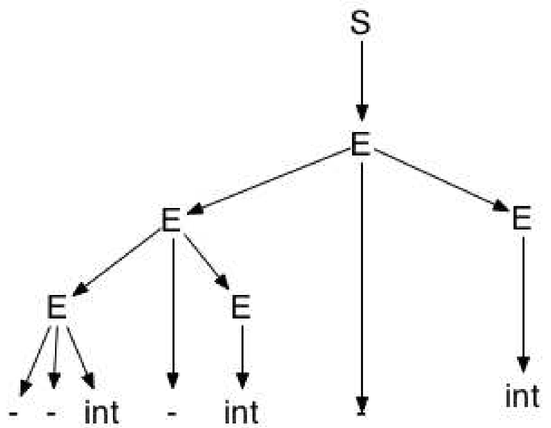
Consider the following grammar with terminals - (the negation operator) and *int*.

S → E
E → E - E | - - E | *int*

(a) Draw all the parse trees for the string ***int - - - int - int*** **[6 points]**

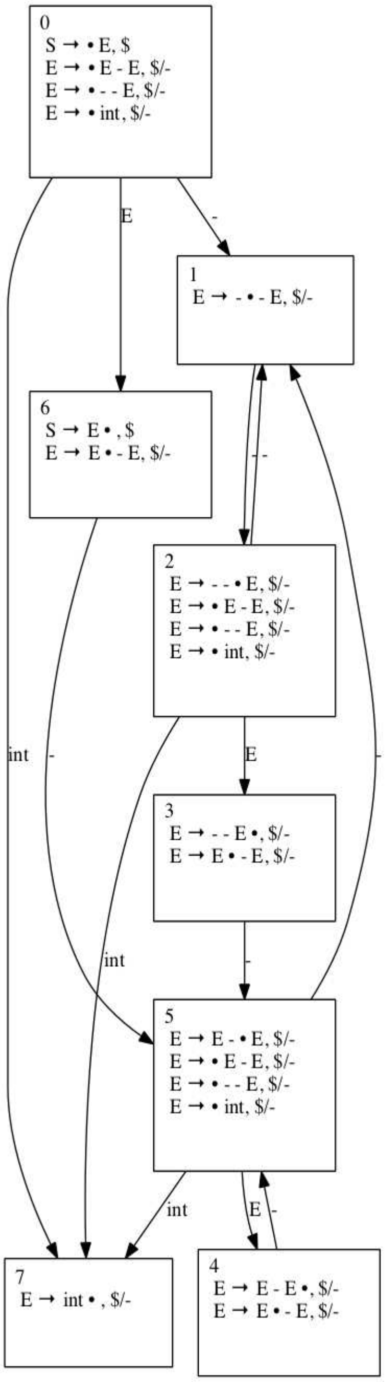






(b) Is this grammar ambiguous? Why or why not? **[2 point]**

The grammar is ambiguous because it admits multiple parse trees for string *int* - - - *int* - *int*



(reduction labels)
state 3: reduce E-> --E on $/-
state 4: reduce E->E-E on $/-
state 6: accept S->E on $
state 7: reduce E->Int on $/-

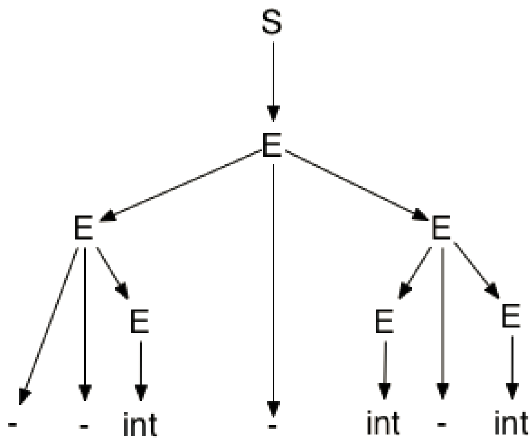
(c) Complete the above partial LR(1) DFA for the grammar. **[16 points]**

- Fill in items of all states by performing closure operation. (6)
- Fill in missing transition labels on all edges (4)
- Write the necessary "reduce by … on … " labels on states (2)
- Add missing transition edges *(Hint: State 2 and State 5)* (4)

(d) For each state with a conflict, list the state, the lookahead token, and the type of conflict (i.e. shift-reduce conflict, or reduce-reduce conflict). **[4 points]**

```
3: Lookahead token: -, shift-reduce conflict
4: Lookahead token: -, shift-reduce conflict
```

Suppose we want the string **- - int - int - int** to have only the following parse tree (call this property P).



(e) Describe in English the precedence and associativity rules necessary to ensure property P. **[4 points]**

1. “-” is right-associative
2. “--” has higher precedence than “-”

(f) Explain, for each conflict in the LR(1) parsing DFA for this grammar, how it should be resolved to ensure property P. **[4 points]**

State 3: reduce instead of shift on “-”
State 4: shift instead of reduce on “-”

(g) Rewrite the grammar to an equivalent unambiguous grammar to ensure property P. Two grammars are equivalent when they accept the same language. **[8 points]**

```
S → E
E → T - E | T
T → - - T | int
```

(h) The Cool grammar has an ambiguity introduced by *let-expression*. Give an example illustrating the ambiguity associated with *let-expression*. **[4 point]**

let … in 1 + 2

can be parsed as:

(let … in 1) + 2

or

let … in (1 + 2)