Name:

Midterm 1

CMSC 430
Introduction to Compilers
Fall 2015

Instructions

This exam contains 7 pages, including this one. Make sure you have all the pages. Write your name on the top of this page before starting the exam.

Write your answers on the exam sheets. If you finish at least 15 minutes early, bring your exam to the front when you are finished; otherwise, wait until the end of the exam to turn it in. Please be as quiet as possible.

If you have a question, raise your hand. If you feel an exam question assumes something that is not written, write it down on your exam sheet. Barring some unforeseen error on the exam, however, you shouldn't need to do this at all, so be careful when making assumptions.

Question	Score	Max
1		25
2		40
3		35
Total		100

Question 1. Short Answer (25 points).

a. (5 points) Briefly explain the difference between a *compiler* and an *interpreter*.

b. (5 points) In the context of lexing and parsing, briefly explain what a *token* is and how it differs from a character in the language to be parsed.

c. (5 points) Briefly explain what makes a language *call-by-value*. Is OCaml a call-by-value language?

d. (10 points) Recall the type of tries from project 1:

```
type ('k, 'v) trie = Trie of 'v option * (('k * ('k, 'v) trie) list)
```

Write a function `find t ks` that returns the value `k` is mapped to in `t`, or aborts with `Not_found` if `ks` is not mapped in `t`. Feel free to use any functions from the `List` module you like.

Question 2. Parsing (40 points).

a. (10 points) Consider the following grammar and associated parsing table.

0.	S'	$\rightarrow$	S
1.	S	$\rightarrow$	aSb
2.	S	$\rightarrow$	D
3.	D	$\rightarrow$	dD
4.	D	$\rightarrow$	d

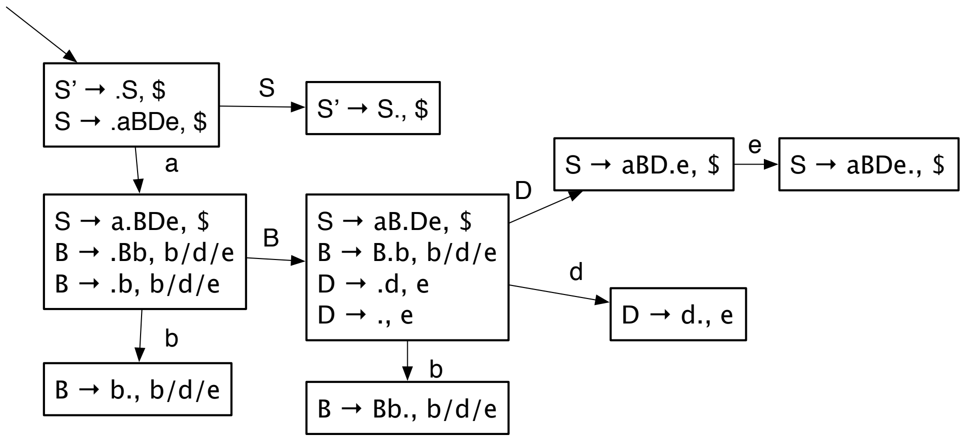
	Action				Goto	
State	a	b	d	$	S	D
0	$s5$		$s3$		1	2
1				acc		
2		$r2$	$r2$			
3		$r4$	$s3$	$r4$		4
4		$r3$		$r3$		
5	$s5$		$s9$		7	6
6		$r2$				
7		$s8$				
8		$r1$		$r1$		
9		$r4$	$s9$			10
10		$r3$				

Fill in the following to show how the string $aadbb\$$ is parsed. You may or may not need to use all the rows. Add extra rows if necessary.

Stack	Input	Action
0	$aadbb\$$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	
	$	

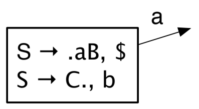
b. (20 points) Draw the LR(1) parsing DFA for the following grammar:

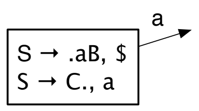
$$\begin{aligned} S &\rightarrow aBDe \\ B &\rightarrow Bb \mid b \\ D &\rightarrow d \mid \varepsilon \end{aligned}$$

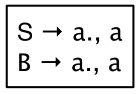


c. (10 points) Draw an LR(1) DFA state, along with any needed outgoing edges, that has a shift/reduce conflict. State on which character a conflict occurs. Similarly, draw an LR(1) DFA state, along with any needed outgoing edges, that has a reduce/reduce conflict, and state on which character a conflict occurs. Be sure to ;abel which state has the shift/reduce conflict and which state has the reduce/reduce conflict.

As an illustration, here is a state without any conflicts:







Question 3. Operational Semantics (35 points.)
a. (10 points) Here are partial big-step operational semantics for arithmetic expressions

$$a ::= n \mid X \mid a + a$$

where $X \in \mathit{Var}$ ranges over variables, and a program state $\sigma : \mathit{Var} \to n$ maps variables to integers n.

$$\textsc{Int}\ \frac{}{\langle n, \sigma \rangle \to n} \qquad \textsc{Var}\ \frac{}{\langle X, \sigma \rangle \to \sigma(X)} \qquad \textsc{Plus}\ \frac{\langle a_1, \sigma \rangle \to n \qquad \langle a_2, \sigma \rangle \to m \qquad p = n + m}{\langle a_1 + a_2, \sigma \rangle \to p}$$

Draw a derivation showing that $\langle 1 + (X + Y), \sigma \rangle \to 6$ if $\sigma = [X \mapsto 2, Y \mapsto 3]$.

b. (8 points) Here are small-step semantics rules for the same language:

$$\textsc{Var}\ \frac{}{X \to_\sigma \sigma(X)} \qquad \textsc{Right}\ \frac{a_2 \to_\sigma a_2'}{a_1 + a_2 \to_\sigma a_1 + a_2'} \qquad \textsc{Left}\ \frac{a_1 \to_\sigma a_1'}{a_1 + n \to_\sigma a_1' + n} \qquad \textsc{Plus}\ \frac{p = n + m}{n + m \to_\sigma p}$$

Show that $1 + (X + Y) \to_\sigma^* 6$ by showing each step of the reduction, where $\sigma = [X \mapsto 2, Y \mapsto 3]$. You don't need to show the derivations that lead to the individual steps, just the steps themselves.

c. (2 points) Does the above small-step semantics evaluate the left- or right-hand side of a summation first? Briefly justify your answer.

d. (10 points) Here I've tried to write down the big-step operational semantics for commands:

$$c ::= \text{skip} \mid X := a \mid c; c \mid \text{if } b \text{ then } c \text{ else } c \mid \text{while } b \text{ do } c$$

But I did a terrible job. List all the mistakes I made, assuming I meant to write a complete, standard semantics for this language. (You don't need to rewrite the semantics, just enumerate my mistakes.)

$$\textsc{Assign}\ \frac{\langle a, \sigma \rangle \to n}{\langle X := a, \sigma \rangle \to \sigma[X \mapsto n]} \qquad \textsc{Seq}\ \frac{\langle c_0, \sigma_1 \rangle \to \sigma_0 \qquad \langle c_1, \sigma \rangle \to \sigma_1}{\langle (c_0; c_1), \sigma \rangle \to \sigma_0}$$

$$\textsc{If-T}\ \frac{\langle b, \sigma \rangle \to \text{true} \qquad \langle c_0, \sigma \rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma \rangle \to \sigma'} \qquad \textsc{If-F}\ \frac{\langle b, \sigma \rangle \to \text{false} \qquad \langle c_0, \sigma \rangle \to \sigma'}{\langle \text{if } b \text{ then } c_0 \text{ else } c_1, \sigma \rangle \to \sigma'}$$

$$\textsc{While-T}\ \frac{\langle b, \sigma \rangle \to \text{true}}{\langle \text{while } b \text{ do } c, \sigma \rangle \to \sigma} \qquad \textsc{While-F}\ \frac{\langle b, \sigma \rangle \to \text{false} \qquad \langle c, \sigma \rangle \to \sigma'}{\langle \text{while } b \text{ do } c, \sigma \rangle \to \sigma'}$$

e. (5 points) Suppose we extend our language with a new form "repeat c until b" that executes c until b becomes true, at which point the loop exits. (Note this means c will always be executed at least once.) Write down *big-step* operational rule(s) for this new form. (Assume there exist other big-step rules of the form $\langle c, \sigma \rangle \to \sigma'$ for commands, and $\langle b, \sigma \rangle \to bv$ for booleans, where bv ranges over true and false.)