Name:

Midterm 1

CMSC 430
Introduction to Compilers
Fall 2016

Instructions

This exam contains 9 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		23
2		42
3		35
Total		100

Question 1. Short Answer (23 points).

a. (5 points) Briefly explain what the *front-end* of a compiler is.

b. (5 points) How are OCaml `ref`s and OCaml records related?

c. (5 points) Briefly explain how LALR(1) parsing and LR(1) parsing are related.

d. (8 points) Recall the following data types from Project 1:

```
type expr =
    EFalse
  | ETrue
  | EVar of string
  | EAnd of expr * expr
  | EOr of expr * expr
  | ENot of expr
  | EForall of string * expr
  | EExists of string * expr

type bvec = expr list (* low order bit at head of list *)
```

Write a function `neq : bvec -> bvec -> expr` that returns an expression representing whether two vectors are **not** equal. You may not call `eq`. You can assume the two vectors are the same length.

Question 2. Parsing (42 points).
a. (12 points) Consider the following grammar and associated parsing table.

0. $S' \rightarrow S$
1. $S \rightarrow SaB$
2. $S \rightarrow c$
3. $B \rightarrow b$
4. $B \rightarrow bB$

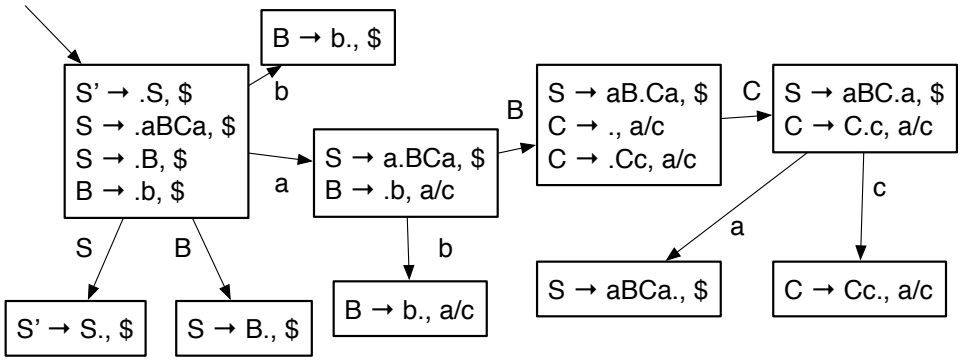
	Action				Goto	
State	a	b	c	\$	S	B
0			$s2$		1	
1	$s3$			acc		
2	$r2$			$r2$		
3		$s4$				7
4	$r3$	$s6$		$r3$		5
5	$r4$			$r4$		
6	$r3$			$r3$		
7	$r1$			$r1$		

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

Stack	Input	Action
0	*cabbab*\$	$s2$
$0, c, 2$	*abbab*\$	$r2$
$0, S, 1$	*abbab*\$	$s3$
$0, S, 1, a, 3$	*bbab*\$	$s4$
$0, S, 1, a, 3, b, 4$	*bab*\$	$s6$
$0, S, 1, a, 3, b, 4, b, 6$	*ab*\$	$r3$
$0, S, 1, a, 3, b, 4, B, 5$	*ab*\$	$r4$
$0, S, 1, a, 3, B, 7$	*ab*\$	$r1$
$0, S, 1$	*ab*\$	$s3$
$0, S, 1, a, 3$	*b*\$	$s4$
$0, S, 1, a, 3, b, 4$	\$	$r3$
$0, S, 1, a, 3, B, 7$	\$	$r1$
$0, S, 1$	\$	acc
	\$	
	\$	

b. (25 points) Draw the LR(1) parsing DFA for the following grammar:

$$\begin{array}{lcl} S & \rightarrow & aBCa \mid B \\ B & \rightarrow & b \\ C & \rightarrow & \varepsilon \mid Cc \end{array}$$



c. (5 points) I had some trouble solving Project 2. At one point, running `ocamlyacc -v` showed the following (partial) output:

1 main : EOF 2 | bexpr EOF 3 bexpr : ID 4 | bexpr AND bexpr	**state 1** %entry% : '\001' . main (5) EOF shift 3 ID shift 4 . error main goto 5 bexpr goto 6	**state 3** main : EOF . (1) . reduce 1
state 4 bexpr : ID . (3) . reduce 3	**state 6** main : bexpr . EOF (2) bexpr : bexpr . AND bexpr (4) EOF shift 7 AND shift 8 . error	**state 7** main : bexpr EOF . (2) . reduce 2
state 8 bexpr : bexpr AND . bexpr (4) ID shift 4 . error bexpr goto 9	9: shift/reduce conflict (shift 8, reduce 4) on AND **state 9** bexpr : bexpr . AND bexpr (4) bexpr : bexpr AND bexpr . (4) AND shift 8 EOF reduce 4	

I need help understanding what's wrong. Write down an input that will cause the parsing DFA to reach state 9 and have to choose between the shift and the reduce that are in conflict. Then, briefly explain what problem with the grammar causes the conflict in that state.

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

$$\begin{array}{rcl} b & ::= & bv \mid X \mid \neg b \mid b \wedge b \mid b \vee b \\ bv & ::= & \text{true} \mid \text{false} \end{array}$$

where $X \in \mathit{Var}$ ranges over boolean values true and false, and a program state $\sigma : \mathit{Var} \to bv$ maps variables to boolean values.

$$\textsc{True}\ \frac{}{\langle \text{true}, \sigma \rangle \to \text{true}} \qquad \textsc{False}\ \frac{}{\langle \text{false}, \sigma \rangle \to \text{false}} \qquad \textsc{Var}\ \frac{}{\langle X, \sigma \rangle \to \sigma(X)}$$

$$\textsc{And}\ \frac{\langle b_1, \sigma \rangle \to bv_1 \qquad \langle b_2, \sigma \rangle \to bv_2 \qquad bv = bv_1 \wedge bv_2}{\langle b_1 \wedge b_2, \sigma \rangle \to bv} \qquad \textsc{Or}\ \frac{\langle b_1, \sigma \rangle \to bv_1 \qquad \langle b_2, \sigma \rangle \to bv_2 \qquad bv = bv_1 \vee bv_2}{\langle b_1 \vee b_2, \sigma \rangle \to bv}$$

Draw a derivation showing that $\langle X \wedge (Y \vee \text{false})), \sigma \rangle \to \text{true}$ if $\sigma = [X \mapsto \text{true}, Y \mapsto \text{true}]$. **Label each step of the derivation with the operational semantics rule used.**

b. (5 points) One big-step rule is missing. Write it down.

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

$$\textsc{Var}\ \frac{\sigma(X) = bv}{X \to_\sigma bv} \qquad \textsc{L-And}\ \frac{b_1 \to_\sigma b_1'}{b_1 \wedge b_2 \to_\sigma b_1' \wedge b_2} \qquad \textsc{T-And}\ \frac{}{\text{true} \wedge b \to_\sigma b} \qquad \textsc{F-And}\ \frac{}{\text{false} \wedge b \to_\sigma \text{false}}$$

$$\textsc{L-Or}\ \frac{b_1 \to_\sigma b_1'}{b_1 \vee b_2 \to_\sigma b_1' \vee b_2} \qquad \textsc{T-Or}\ \frac{}{\text{true} \vee b \to_\sigma \text{true}} \qquad \textsc{F-Or}\ \frac{}{\text{false} \vee b \to_\sigma b}$$

Show that $X \wedge (Y \vee \text{false})) \to_\sigma^* \text{true}$ if $\sigma = [X \mapsto \text{true}, Y \mapsto \text{true}]$. Show each step of the reduction, but you don't need to show the derivations that lead to the individual steps, just the steps themselves. (The relation $b \to_\sigma^* b'$ means b reaches b' in zero or more steps of $\to_\sigma$.)

d. (7 points) Suppose we extend the grammar to include implication: $b ::= \ldots \mid b \Rightarrow b$. Write down the corresponding small-step rules, following the pattern in part c, using the normal interpretation of logical implication.

e. (5 points) Briefly explain what a *normal form* is in small-step operational semantics. What are the normal form(s) of the semantics in part c?