03-60-214: Midterm Exam (2014)

Last Name	First Name	Student ID	Marks

This is close-book exam. You can use both pen or pencil. If you need more space, you can write on the back of the paper. There are 8 pages. The total score is 85, and will be scaled back to 20%.

1. (10 points) The following JavaCup specification is supposed to count the number of methods in a program written in a very small language that is a subset of our Tiny language. This language has only assignment statements. In the following JavaCup file, $assignmentStatementList$ is a list of assignment statements, each ends with a semicolon, just the same as our Tiny language. Suppose that the symbols in uppercase letters are terminals, and they are handled by the scanner correctly.

   (a) Fill in the blanks for the missing parts, so that it is a correct JavaCup specification;

   (b) Write a simple program that can be accepted be this grammar.

```
------------        PLUS, MINUS, TIMES, DIVIDE, LPAREN, RPAREN;
------------        program, method;

------------        ----------------------------------------;

------------        ----------------------------------------;

precedence left PLUS, MINUS;
precedence left TIMES, DIVIDE;

program ::= method  {: ---------------------  :}

        | method:e1 program:e2  {:  ---------------------- :} ;

method ::= INT ID LPAREN RPAREN BEGIN assignmentStatementList END;

assignmentStatementList ::= ----------------------------------------

                    |  ---------------------------------------

assignmentStatement ::= ID EQUALSIGN expr;
expr       ::= expr PLUS expr    | expr MINUS expr
             | expr TIMES expr    | expr DIVIDE expr
             | LPAREN expr RPAREN  | NUMBER | ID
             ;
```

2. (a) (5 points) Write the algorithm (pseudo code) to generate the closure of LR(1) items. Note that you should write the closure algorithm for LR(1), not for other parsing methods. The following gives you a hint on the structure of the algorithm.

Algorithm 1: CLOSURE(I) (dragon book p.261.

Input: I, which is a set of LR(1) items
Output: I, which has more LR(1) items

```
repeat
    for each item [A → α • Bβ, a] do
        for each production B → γ in G do
            for each terminal b in First(βα) do
                add [B → •γ, b] to I ;
            end
        end
    end
until I is not changed;
return I;
```

(b) (5 points) Given the following grammar for arithmetic expression.

$$E \rightarrow E + E \tag{1}$$
$$E \rightarrow E * E \tag{2}$$
$$E \rightarrow (E) | id \tag{3}$$

Applying the above algorithm, write the LR(1) closure for the first state, i.e., for item

$$E' \rightarrow \bullet E, \$$$

Note that you should write the LR(1) closure, not SLR or LR(0) closure.

(c) (5 points) From the first state, draw a partial transition LR(1) diagram until it contains a state that has Shift/Reduce conflict.

3. (10 points) The following grammar for If statement is ambiguous. Rewrite the grammar to remove the ambiguity, by enforcing that $else$ always matches with the closest $then$. Note that this grammar is a simplified version in our assignment, in that there are only $ifStmt$ and $assignmentStmt$.

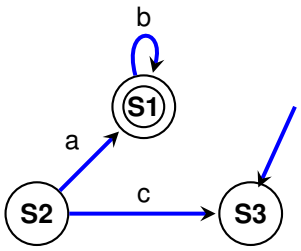
$$ifStmt \rightarrow if\ expr\ then\ stmt \tag{4}$$
$$|if\ expr\ then\ stmt\ else\ stmt \tag{5}$$
$$stmt \rightarrow ifStmt \tag{6}$$
$$|assignmentStmt \tag{7}$$

4. (10 points) Give short answers to the following questions:

   (a) (2 points) In the LR parsing method,

   - L stands for _____, and
   - R stands for_____ .

   (b) (2 points) Given the following transition diagram. Write the corresponding regular expression.



   (c) (2 points) Given the following grammar, where `A` is a non-terminal, `a` and `b` are terminals:

$$A \rightarrow aAb|\epsilon$$

   Is it possible to have an equivalent regular expression? If yes, write the corresponding regular expression.

   (d) (2 points) Given the regular expression `(ab)+`. Is it possible to have an equivalent regular grammar. If yes, write the corresponding regular grammar.

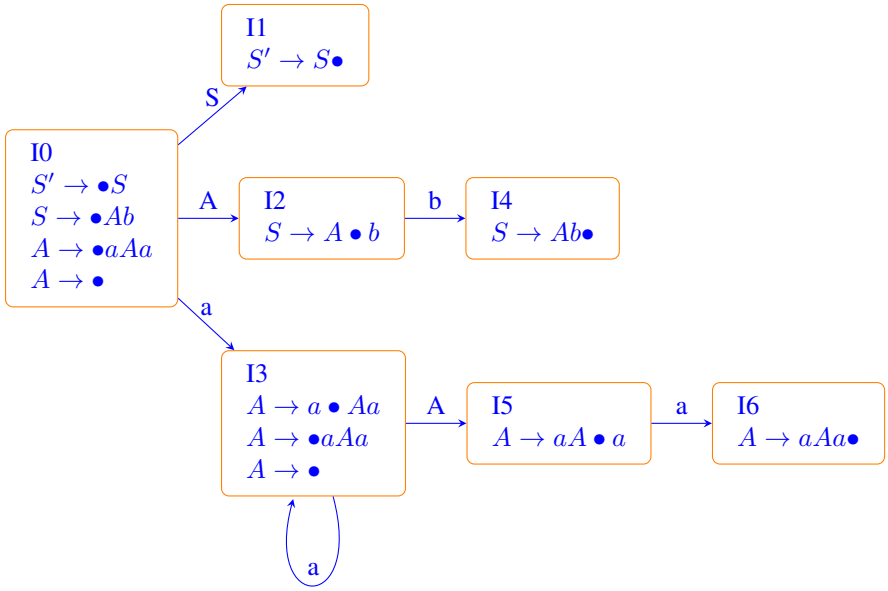
   (e) (2 points) Write a grammar for the all possible palindromes over alphabet {`a, b`}, i.e., the strings read the same backward and forward, such as `bb`, `bab`, `aabbabbaa`.

5. (10 points) Given the following grammar where `S, A` are non terminals and `a, b` are terminals.

$$S \to Ab \quad (1)$$
$$A \to aAa \quad (2)$$
$$A \to \epsilon \quad (3)$$

(a) Draw the LR(0) transition diagram;

(b) Draw the LR(0) parsing table.

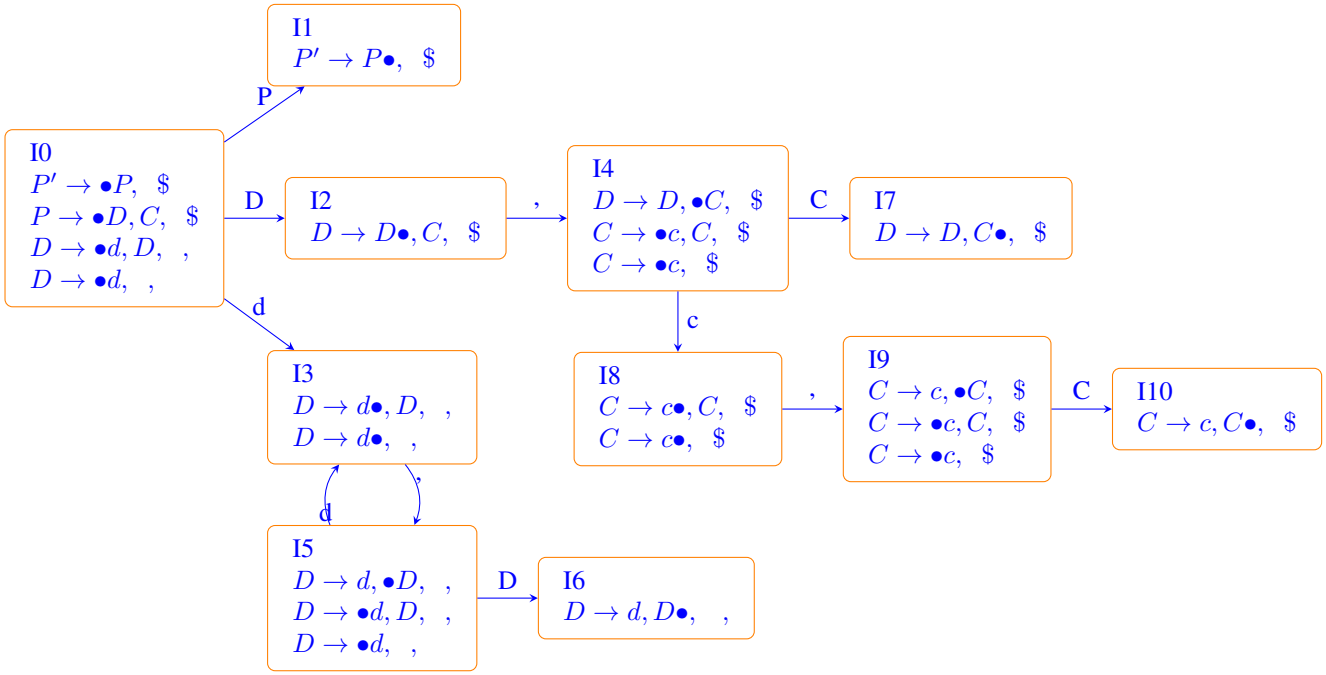


State	Action			Goto	
	a	b	$	S	A
S0	S3/R3	R3	R3	1	2
S1			Accept		
S2		S4			
S3	R3/S3	R3	R3		5
S4	R1	R1	R1		
S5	S6	S6	S6		
S6	R2	R2	R2		

6. (10 points) Consider the following grammar, where P, C, and D are non-terminals and other symbols are terminals.

$$P' \to P$$
$$P \to D, C$$
$$D \to d, D|d$$
$$C \to c, C|c$$

(a) (8 points) Construct the entire LR(1) transition diagram. Note that you must write the LR(1) diagram, not LR(0) or SLR.



(b) (2 points) Is it an LR(1) grammar? Give a brief explanation.

It is not an LR(1) grammar, because there is a shift reduce conflict in state I3.

7. (10 points) Write the LR parsing algorithm. Note that the parsing algorithm is the same for variants of LR parsing tables. Suppose that the $nextToken()$ method returns the next token.

Algorithm 2: LR Parsing (dragon book p. 251)

Input: An input string w, an LR-parsing table that consists of Action and Goto sub-tables, and a grammar G;
Output: The reduction steps of w if w is in the language of G; otherwise an error indication.

```
The initial state s_0 is in stack;
a=nextToken() ;
while true do
    s= the top state on the stack;
    if ACTION[s,a]=shift t then
        push t onto the stack;
        a=nextToken();
    end
    else if ACTION[s,a]=reduce A → β then
        pop |β| states off the stack;
        t=the top state in the stack;
        push GOTO[t,A] onto the stack;
    end
    else if ACTION[s,a]=accept then
        break;
    end
    else
        call error-procedure;
    end
end
return I;
```

8. (10 points) Given the following grammar and parse table, write the parse trace for the string `aabb` in the following table. You can add more rows if needed, or leave some rows empty if no more steps are required.

$$
\begin{aligned}
S' &\to S && (0)\\
S &\to AA && (1)\\
A &\to aA && (2)\\
A &\to b && (3)
\end{aligned}
$$

Stack of states	Remaining input	Action
S0$	aabb$	

the same as the slides.

9. (A question from Midterm 1)

Algorithm 3: Simulate DFA (dragon book p. 151)

Input: An input string x, a DFA with start state s_0, accepting states F.
Output: "yes" if D accepts x, "no" otherwise.
$s = s_0$;
while *(c=nextChar())!=eof* **do**
    s= move(s,c);
end
if $s \in F$ **then**
    return "yes";
end
return "no";