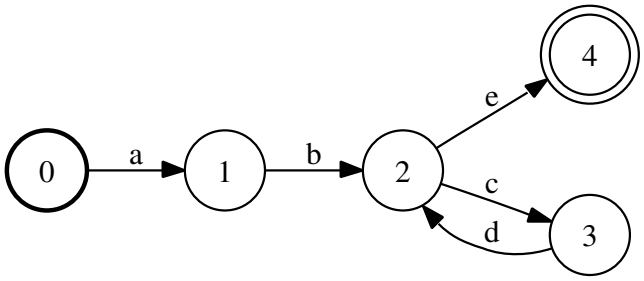
CMPT 379 - Summer 2016 - Midterm

(1) (6pts) Lexical Analysis:

a. Consider the following DFA:



Provide a regular expression for the regular language generated by this DFA.

Answer: `ab(cd)*e`

b. You are given the following ordered list of token definitions:

`TOKEN_A` $(ab)^*a$
`TOKEN_B` $(ab)^*a(ca)^*$
`TOKEN_C` $bab(bab)^*$
`TOKEN_D` $a^*ba(ba)^*$

Provide the tokenized output for the input string *abacababababa* using the greedy longest match lexical analysis method.

Answer:
First match:

`TOKEN_A` *aba*
`TOKEN_B` *abaca*
`TOKEN_C` no match
`TOKEN_D` *aba*

Second match:

`TOKEN_A` no match
`TOKEN_B` no match
`TOKEN_C` *bab*
`TOKEN_D` *bababa*

Output:

`TOKEN_B` *abaca*
`TOKEN_D` *bababa*

c. The following CFG generates a regular language. Provide a regular expression that generates the same language as this CFG.

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow c \mid \epsilon \\ B &\rightarrow cbB \mid ca \end{aligned}$$

Answer:
`c?(cb)*ca`

(2) (7pts) Consider the following context-free grammar G.

$$S' \rightarrow S \quad (1)$$
$$S \rightarrow aSb \quad (2)$$
$$S \rightarrow \epsilon \quad (3)$$

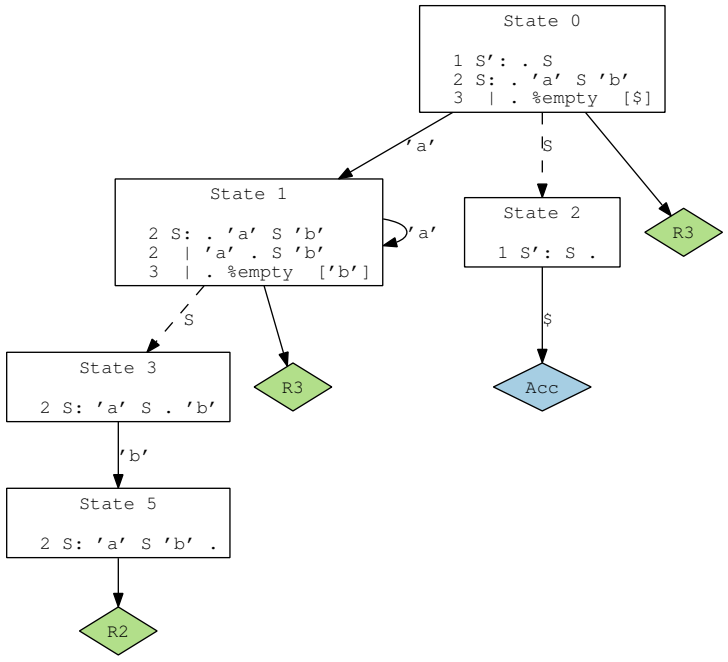
a. Provide FIRST(S).

Answer:

$$\text{FIRST}(S) = \{a, \epsilon\}$$

b. Provide the LR(0) itemset automaton for G.

Answer:



c. Is G LR(0)? If G is not LR(0), provide the shift/reduce or reduce/conflicts to justify your answer.

Answer: G is not LR(0) because of two shift/reduce conflicts. State 0 has a shift on a and reduce of $S \rightarrow \epsilon$. Same shift/reduce conflict in State 1.

d. Is G SLR(1)? Justify your answer using FOLLOW(S).

Answer: G is SLR(1) because in each of the two shift/reduce conflicts we can reduce $S \rightarrow \epsilon$ using the FOLLOW set of S which is $\{b, \$\}$.

(3) (7pts) Consider the following incomplete yacc program where each non-terminal is declared to have an integer type.

```
S : E   { print_int($1); }
E : "0" { $$ = 0;  }
  | D1  { $$ = $1; }
  | D1 D2 {
      if ($2 == 1) { $$ = $1 * 10 + 10; }
      else { $$ = $1 * 10; }
    }
  ;
D2 : L { $$ = $1; }
   | G { $$ = $1; }
   ;
```

Some non-terminals appear in the right hand side of some rules above but their definitions are missing. The following testcases indicate the task to be solved by the complete yacc program.

Input	**Output**
0	0
5	5
9	9
02	syntax error
10	10
22	20
78	80
35	40
96	100
100	syntax error

Complete the yacc program by providing the missing non-terminal definitions so that the yacc program can pass all the testcases. You *must* follow the requirements below:

- Provide both the context-free rules for each non-terminal and the action for each rule.
- You can only add new definitions; you cannot change the rules or actions in the yacc program above.
- You cannot add new rules for non-terminals already defined above (you cannot add rules or change actions for non-terminals `S`, `E` and `D2`).
- Your new definitions cannot introduce new non-terminals.

Answer:

```
D1 : "1" { $$ = 1; }
   | "2" { $$ = 2; }
   | "3" { $$ = 3; }
   | "4" { $$ = 4; }
   | "5" { $$ = 5; }
   | "6" { $$ = 6; }
   | "7" { $$ = 7; }
   | "8" { $$ = 8; }
   | "9" { $$ = 9; }
   ;

L : "0" { $$ = 0; }
   | "1" { $$ = 0; }
   | "2" { $$ = 0; }
   | "3" { $$ = 0; }
   | "4" { $$ = 0; }
   ;

G : "5" { $$ = 1; }
   | "6" { $$ = 1; }
   | "7" { $$ = 1; }
   | "8" { $$ = 1; }
   | "9" { $$ = 1; }
   ;
```