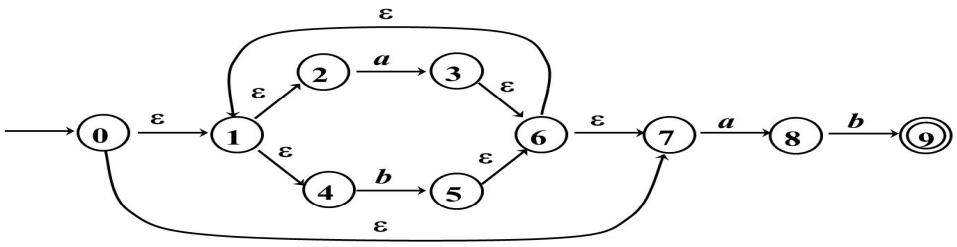
1. Convert the following NFA to DFA (minimum-state DFA, using state minimization algorithm). You need to show the conversion steps. (10 cents)



2. Given the grammar

A → bAaA

A → aAbA

A → ε

a) Give out the leftmost derivation of the string *baba.* (3 cents)
b) Is the grammar ambiguous? Why? (5 cents)
c) What's the language described by the grammar? (2 cents)

3. Consider the following grammar

D → T L
T → int | real
L → id R
R → , id R | ε

a). Calculate FIRST and FOLLOW set for non-terminals in the grammar, and show the nullable nonterminals. (10 cents)
b). Construct the LL(1) parsing table for the grammar. Is the grammar a LL(1) grammar? Why? (15 cents)

4. The program 4.1 in the page 89 has presented the recursive-descent interpreter for part of the Grammar 3.15 (see page 53). Please continue to finish the recursive-descent procedure for the remaining part of Grammar 3.15. (10 cents)

5. Given the program 5.2 in the page 106.
a) Please describe how to use the hash table to maintain the scope of variables
b) Improve the hash implementation to hide the representation of the *table* type inside an abstract module, so that clients are not tempted to manipulate the data structure directly(only through the *insert*, *lookup*, and *pop* operation)

6. Please answer the following questions.
a) Whether we can use stacks to hold all local variables for higher-order functions or not? Why? (5 cents)
b) Describe the function of the following four frame interfaces which are defined in page 136. (10 cents)

7. Implement the following methods defined in page 156 by C language.
a) static T_stm unNx (Tr_exp e) (7 cents)
b) static struct Cx unCx (Tr_exp e) (8 cents)