CS-375: Introduction to Compilers
Solutions for Quiz 1.

1. (5 points) NFA

b) (2 points for correct first state, ½ point per step.) States: On input **abababb**, we get the following path:

<table>
<thead>
<tr>
<th>Input</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0</td>
</tr>
<tr>
<td>A</td>
<td>0,1,2</td>
</tr>
<tr>
<td>B</td>
<td>0,3,5</td>
</tr>
<tr>
<td>A</td>
<td>0,1,2</td>
</tr>
<tr>
<td>B</td>
<td>0,3,5</td>
</tr>
<tr>
<td>A</td>
<td>0,1,2</td>
</tr>
<tr>
<td>B</td>
<td>0,3,5</td>
</tr>
<tr>
<td>b</td>
<td>0,4</td>
</tr>
</tbody>
</table>

c) (10 points) DFA
2. **[8 points for LALR(1) look-ahead explanation]** LALR(1):
   It is LALR(1) since look-ahead for A->d in state q4 is \{a\} and look-ahead for A->d in state q7 is \{c\}

   **[6 points for LR(0) automaton]**

   ![LALR(1) Automaton Diagram]

   **[6 points for shift/reduce conflicts]** SLR(1): Follow(A) = \{a,c\}. So grammar is not SLR(1) since the use of Follow sets does not resolve shift/reduce conflicts in states q4 and q7.

3. 

   a) **[7 points]** F-sets: F(S) = \{S,A,B\}, F(A)=\{A,B\}, F(B)=\{A,B\}
   b) **[16 points]** Algorithm
      
      While F changes for any non-terminal
      
      For each non terminal N
      
      Add N to F(N) (2 point)
      
      For each production p that starts with non-terminal X
      
      Add X to F(N) (2 point)
      
      Add F(X) to F(N) (4 point)
      
      If X is nullable (8 point)
      
      Add F(trailing part of production) to F(N)

   c) **[7 points]** Sample run.
      
      F(S)=\{\}. F(A)=\{\}. F(B)=\{\}
      
      1) F(S)=\{S,A\}, F(A)=\{B\}, F(B)=\{A,B\}
      
      2) F(S)=\{S,A,B\}, F(A)=\{A,B\}, F(B)=\{A,B\}

4. 

   a) **[3 points]** LL(1) : It is LL(1) because you can disambiguate based on a look-ahead of one symbol . First(A) = \{\epsilon\}, First(B)=\{\epsilon\}, Follow(A)=\{a,b\}, Follow(B)=\{a,b\}. The look-ahead for S->AaAb is \{a\}, and for S->BbBa the look-ahead is \{b\}. So the grammar is LL(1).
   
   b) **[3 points]**
c) [4 points] **SLR(1):** Not SLR(1). LR(0) automaton has a reduce-reduce conflict in state q0. The Follow sets for A and B, \( L(A) = \{a,b\} \), and \( L(B) = \{b,a\} \), hence the SLR(1) parser cannot decide which production to apply based on the look-ahead symbols.

5.

a) [4 points] Fortran, John Backus, IBM, 1950s-1960s

b) [1 points] Backus Normal Form or Backus Naur Form (later due to Peter Naur).

c) [1 points] False, it does accept ambiguous grammars but requires precedence rules to be specified (explicitly or implicitly)

d) [2 points] \( 2^n \), because we have to consider all possible subsets.

e) [2 points] LALR(1) is a strict superset of LR(0) since LR(0) require no look-ahead.

f) [2 points] True

g) [2 points] CUP, JavaCC, ANTLR

h) [1 points] Compute Follow and First sets.

i) [2 points] 0-address are stack based machines, and 3 address have instructions of the format \( \text{res} = \text{operand1} \text{ operator} \text{ operand2} \)

j) [2 points] Simple, makes expression evaluation easy and requires less temporaries. Not suitable for register based machines.

k) [1 points] Stack frame is a section of the stack used for a function/procedure invocation. It stores all the context information for a particular invocation such as parameters, return address, return stack address, etc.