1. (20 pts) LR parsing
   a. (10 pts) Given the following ACTION/GOTO table, show the parse of the string “cc”. Describe the stack, remaining input, and action performed at each step.

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Goto</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Shift 3</td>
<td>Shift 1</td>
</tr>
<tr>
<td>1</td>
<td>Reduce S → c</td>
<td>Reduce S → c</td>
</tr>
<tr>
<td>2</td>
<td>Shift 3</td>
<td>Shift 4</td>
</tr>
<tr>
<td>3</td>
<td>Reduce S → Sa</td>
<td>Reduce S → Sa</td>
</tr>
<tr>
<td>4</td>
<td>Reduce S → Sc</td>
<td>Reduce S → Sc</td>
</tr>
</tbody>
</table>

   
<table>
<thead>
<tr>
<th>Stack</th>
<th>Remaining Input</th>
<th>Action</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>$0$</td>
<td>c c $</td>
<td>Shift 1</td>
<td></td>
</tr>
<tr>
<td>$0$ c 1</td>
<td>c $</td>
<td>Reduce S → c</td>
<td></td>
</tr>
<tr>
<td>$0$ S</td>
<td>c $</td>
<td>Goto 2</td>
<td>Intermediate Step</td>
</tr>
<tr>
<td>$0$ S 2</td>
<td>c $</td>
<td>Shift 4</td>
<td></td>
</tr>
<tr>
<td>$0$ S 2 c 4</td>
<td>$</td>
<td>Reduce S → Sc</td>
<td></td>
</tr>
<tr>
<td>$0$ S</td>
<td>$</td>
<td>Goto 2</td>
<td>Intermediate Step</td>
</tr>
<tr>
<td>$0$ S 2</td>
<td>$</td>
<td>Accept</td>
<td></td>
</tr>
</tbody>
</table>

   b. (6 pts) Consider the following set of LR(1) items in the state of a LR(1) parser. Find all shift/reduce and reduce/reduce errors, and list the LR(1) items and lookaheads causing the conflicts.

   [A → b • B, a ]
   [A → • a, b ]
   [A → bb •, c ]
   [B → • b, a ]
   [B → • bbA, a ]
   [B → b •, a ]

   Shift/reduce error for LR(1) items [A → • a, b ] [B → b •, a ] on lookahead a.

   c. (4 pts) What is the effect on the associativity of an operator if a shift/reduce conflict for the operator is resolved by always performing shift? Explain why.

   **Operator becomes right associative, since (left) operator on stack will be reduced after (right) operator in remaining input.**
2. (40 pts) LR(1) parsing
Consider the following grammar: $S' \rightarrow S \quad S \rightarrow aSb \mid a$

a. (22 pts) Compute the canonical set of LR(1) items

```
State 0
[S' → • S, $]
[S → aSb, $]
[S → • a, $]
```

```
State 2
[S → aS, $]
[S → a$, $]
[S → • aSb, b]
[S → • a, b]
```

```
State 3
[S → aS • b, $]
```

```
State 4
[S → aSb •, $]
```

```
State 5
[S → • aSb, b]
[S → • a, b]
```

```
State 6
[S → • aSb, b]
```

```
State 7
[S → aSb •, b]
```

```
State 1
[S' → S •, $]
```

```
State 0
[S' → • S, $]
[S → aSb, $]
[S → • a, $]
```

```
State 2
[S → aS, $]
[S → a$, $]
[S → • aSb, b]
[S → • a, b]
```

```
State 3
[S → aS • b, $]
```

```
State 4
[S → aSb •, $]
```

```
State 5
[S → • aSb, b]
[S → • a, b]
```

```
State 6
[S → • aSb, b]
```

```
State 7
[S → aSb •, b]
```

```
State 1
[S' → S •, $]
```

b. (12 pts) Construct the LR(1) ACTION/GOTO table

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Goto</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Shift 2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Accept</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Shift 5</td>
<td>Reduce $S \rightarrow a$</td>
</tr>
<tr>
<td>3</td>
<td>Shift 4</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Reduce $S \rightarrow aSb$</td>
</tr>
<tr>
<td>5</td>
<td>Shift 5</td>
<td>Reduce $S \rightarrow a$</td>
</tr>
<tr>
<td>6</td>
<td>Shift 7</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reduce $S \rightarrow aSb$</td>
<td></td>
</tr>
</tbody>
</table>

c. (6 pts) Prove the grammar is or is not LALR(1)
Following is set of canonical items produced when merging states with identical cores. No reduce/reduce conflicts added, grammar is LALR(1).