1. (2 pts) What is the output (if any) of the following Ruby program? Write FAIL if code does not execute. Recall that Array.collect applies a code block to each element of an array, and creates a new array from the value returned by the code block.

```ruby
a = [1,2,3]    # Output = 0!
b = a.collect { |x| "#{x-1}!" } 1!
puts b 2!
```

2. (6 pts) Consider the regular expression ab*. Recall * has higher precedence than concatenation.
   a. (1 pt) Does it accept the string “a”? Circle one: Yes No
   b. (5 pts) Create a NFA for ab*, using the algorithm discussed in class.

3. (12 pts) Consider the following NFA.
   a. (2 pts) Does it accept the string “aa”? List a possible sequence of state transitions (e.g., 1,3,4) leading to acceptance / rejection of “aa”.
      
      Yes: 1,2,4,1,2,4
   b. (10 pts) Convert the NFA to a DFA using the subset construction algorithm discussed in class. Be sure to label each state in the DFA with the corresponding state(s) in the NFA.

<table>
<thead>
<tr>
<th>NFA</th>
<th>DFA resulting from subset construction</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="NFA" /></td>
<td><img src="image2.png" alt="DFA" /></td>
</tr>
</tbody>
</table>