1. (a) $bb$ is accepted. Computations on input $bb$:
   $(s, bb), (s, b), (s, \varepsilon)$
   $(s, bb), (s, b), (p, b), (r, \varepsilon)$
   $(s, bb), (p, bb), (r, b)$ [stuck]

   (b) $ab$ is accepted. Computations on input $ab$:
   $(s, ab), (p, ab), (p, b), (r, \varepsilon)$
   $(s, ab), (p, ab), (r, b)$ [stuck]

   (c) $aba$ is not accepted. Computations on input $aba$:
   $(s, aba), (p, aba), (p, ba), (r, a), (s, \varepsilon)$
   $(s, aba), (p, aba), (p, a), (s, \varepsilon), (p, \varepsilon)$
   $(s, aba), (p, aba), (r, ba)$ [stuck]

   (d) $\varepsilon$ is not accepted. Computations on input $\varepsilon$:
   $(s, \varepsilon)$
   $(s, \varepsilon), (p, \varepsilon)$

2. The new states are $\emptyset$, $\{s\}$, $\{p\}$, $\{r\}$, $\{s, p\}$, $\{s, r\}$, $\{p, r\}$, $\{s, p, r\}$. The initial state is $\{s, p\}$. The favourable states are $\{r\}$, $\{p, r\}$, $\{s, r\}$, and $\{s, p, r\}$. The equivalent DFA is:

   After removing the unreachable states, we obtain:

3. Equivalent DFAs are shown in the picture below:
4. An equivalent DFA is shown below

(observe that the automaton accepts all words starting with \(a\))

5. Transition table:

<table>
<thead>
<tr>
<th>subset</th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(∅)</td>
<td>(∅)</td>
<td>(∅)</td>
<td>(∅)</td>
</tr>
<tr>
<td>(q_1)</td>
<td>(q_1)</td>
<td>(q_1)</td>
<td>(q_1)</td>
</tr>
<tr>
<td>(q_2)</td>
<td>(q_2)</td>
<td>(q_2)</td>
<td>(q_2)</td>
</tr>
<tr>
<td>(q_3)</td>
<td>(q_3)</td>
<td>(q_1, q_2)</td>
<td>(q_2, q_3)</td>
</tr>
<tr>
<td>(q_1, q_2)</td>
<td>(q_2, q_3)</td>
<td>(q_2, q_3)</td>
<td>(∅)</td>
</tr>
<tr>
<td>(q_2, q_3)</td>
<td>(q_3)</td>
<td>(q_1, q_2, q_3)</td>
<td>(q_2, q_3)</td>
</tr>
<tr>
<td>(q_1, q_3)</td>
<td>(q_1, q_2)</td>
<td>(q_1, q_2, q_3)</td>
<td>(q_2, q_3)</td>
</tr>
<tr>
<td>(q_1, q_2, q_3)</td>
<td>(q_2, q_3)</td>
<td>(q_1, q_2, q_3)</td>
<td>(q_2, q_3)</td>
</tr>
</tbody>
</table>
6. The languages are different in both cases. Indeed,

\[ abb \in L((aa \cup ab)^*) \text{ but } abb \notin L((aa \cup ab)^*) \]

and

\[ abb \in L((ba \cup ab)^*) \text{ but } abb \notin L((ba \cup ab)^*). \]

7. – An arbitrary number of binary characters (0 or 1) precedes the substring 01 or all words built of 0 and 1 that end with 01;
– strings that must contain a zero but which otherwise consist only of ones;
– strings of even length consisting only of ones;
– an arbitrary number of repetitions of a string consisting two 1’s and an arbitrary number of zeroes in arbitrary positions or all strings built of 0 and 1 with even number of 1’s;
– strings containing the substring 01;
– strings of the form 111...000..., that is, strings that begin with zero or more ones followed by zero or more zeroes;
– \((10 \cup 0)^*\) is all strings that do not contain the substring 11; \((1 \cup 10)^*\) is all strings that do not contain the substring 00; so the concatenation of these is all strings where each occurrence of 00 precedes all occurrences of 11 (in other words, no 11 occurs after 00).
– all strings that do not contain the substring 101.

8. (a) \(a(a \cup b)^*b(a \cup b)^* \text{ or } aa^*b(a \cup b)^*\)
(b) \((bb)^*(aa)^* \cup (bb)^*ba(aa)^*\)
(c) \((aa \cup bb \cup (ab \cup ba)(aa \cup bb)^*(ab \cup ba))^*\)
(d) A string that does not have both the substrings \(bba\) and \(abb\) is a string that does not contain \(bba\) or does not contain \(abb\). The language of all strings that do not contain \(bba\) can be defined by the regular expression \(a^*(bba^*)^*b^*\). The language of all strings that do not contain \(abb\) can be defined by the regular expression \(b^*(a^*ab)^*a^*\). This gives the following answer:

\((a^*(bba^*)^*b^*) \cup (b^*(a^*ab)^*a^*)\).
9. (a) DFA:

![DFA Diagram](image)

(b) Note that $L((aa \cup ab^*)^*) = L((ab^*)^*) = L[\varepsilon \cup a(a \cup b)^*]$. This gives the following DFA:

![DFA Diagram](image)

10.
11. (a) NFA:

DFA:

(b) NFA:
DFA: