Practice Questions on NFA

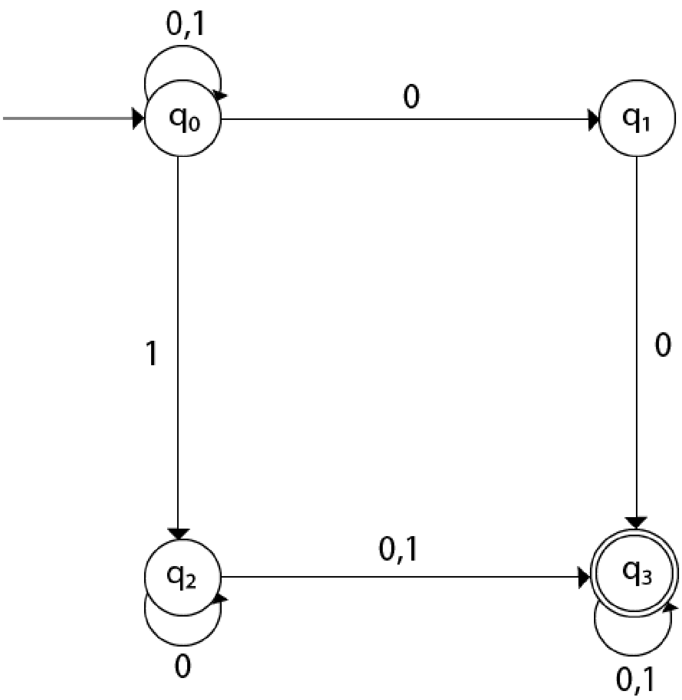
Example 1:

Design an NFA for the transition table as given below:

Present State	0	1
→q0	q0, q1	q0, q2
q1	q3	ε
q2	q2, q3	q3
→q3	q3	q3

Solution:

The transition diagram can be drawn by using the mapping function as given in the table.



Here,

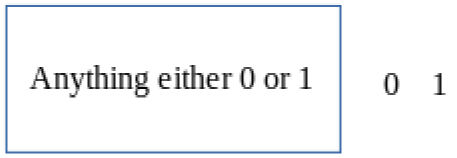
1. δ(q0, 0) = {q0, q1}
2. δ(q0, 1) = {q0, q2}

3. Then, δ(q1, 0) = {q3}
4. Then, δ(q2, 0) = {q2, q3}
5. δ(q2, 1) = {q3}
6. Then, δ(q3, 0) = {q3}
7. δ(q3, 1) = {q3}

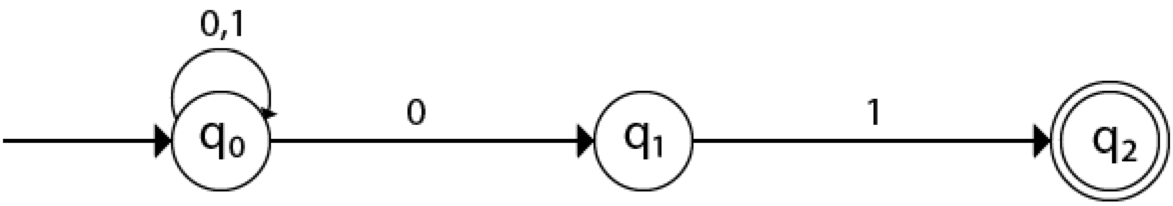
Example 2:

Design an NFA with ∑ = {0, 1} accepts all strings ending with 01.

Solution:



Hence, NFA would be:

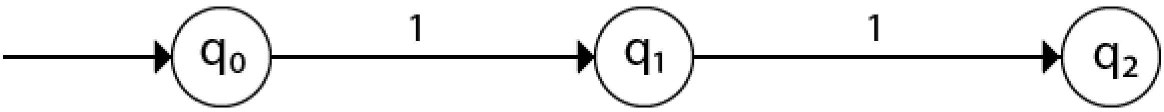


Example 3:

Design an NFA with ∑ = {0, 1} in which double '1' is followed by double '0'.

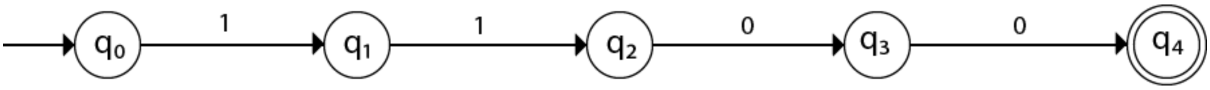
Solution:

The FA with double 1 is as follows:



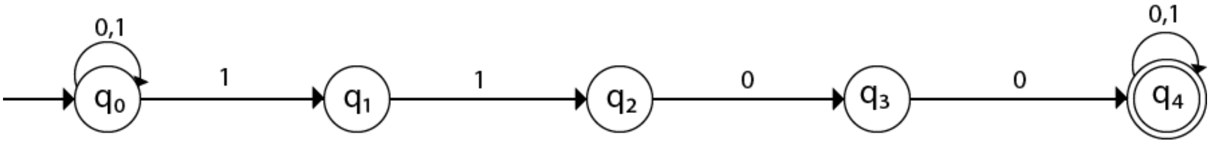
It should be immediately followed by double 0.

Then,



Now before double 1, there can be any string of 0 and 1. Similarly, after double 0, there can be any string of 0 and 1.

Hence the NFA becomes:



Now considering the string 01100011

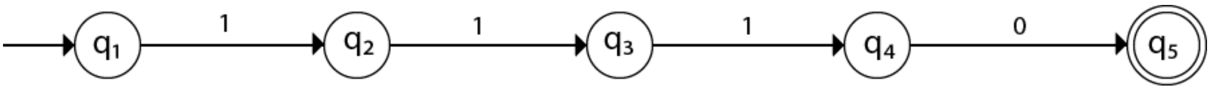
1. q0 → q1 → q2 → q3 → q4 → q4 → q4 → q4

Example 4:

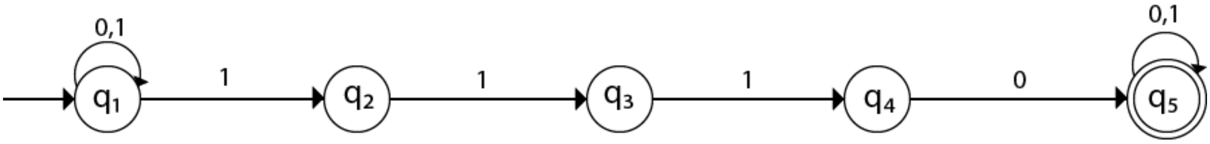
Design an NFA in which all the strings contain a substring 1110.

Solution:

The language consists of all the strings containing substring 1010. The partial transition diagram can be:



Now 1010 could be the substring. Hence, we will add the inputs 0's and 1's so that the substring 1010 of the language can be maintained. Hence, the NFA becomes:



The transition table for the above transition diagram can be given below:

Present State	0	1
→q1	q1	q1, q2
q2		q3
q3		q4
q4	q5	
*q5	q5	q5

Consider a string 111010,

1. δ(q1, 111010) = δ(q1, 1100)
2. = δ(q1, 100)
3. = δ(q2, 00)

Got stuck! As there is no path from q2 for input symbol 0. We can process string 111010 in another way.

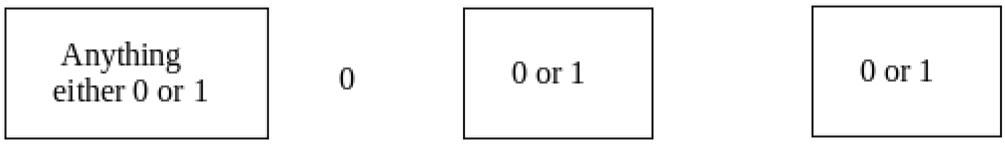
1. δ(q1, 111010) = δ(q2, 1100)
2. = δ(q3, 100)
3. = δ(q4, 00)
4. = δ(q5, 0)
5. = δ(q5, ε)

As state q5 is the accept state. We got the complete scan, and we reached the final state.

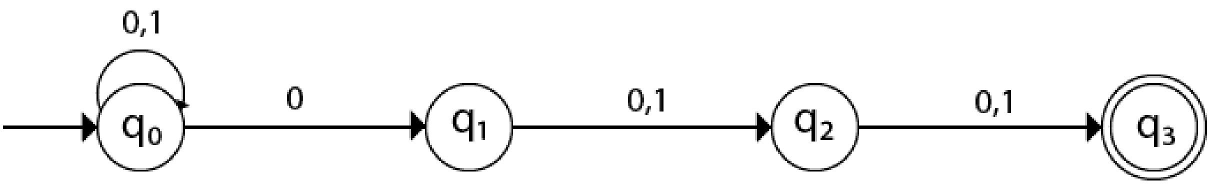
Example 5:

Design an NFA with ∑ = {0, 1} accepts all string in which the third symbol from the right end is always 0.

Solution:



Thus we always get the third symbol from the right end as '0'. The NFA can be:

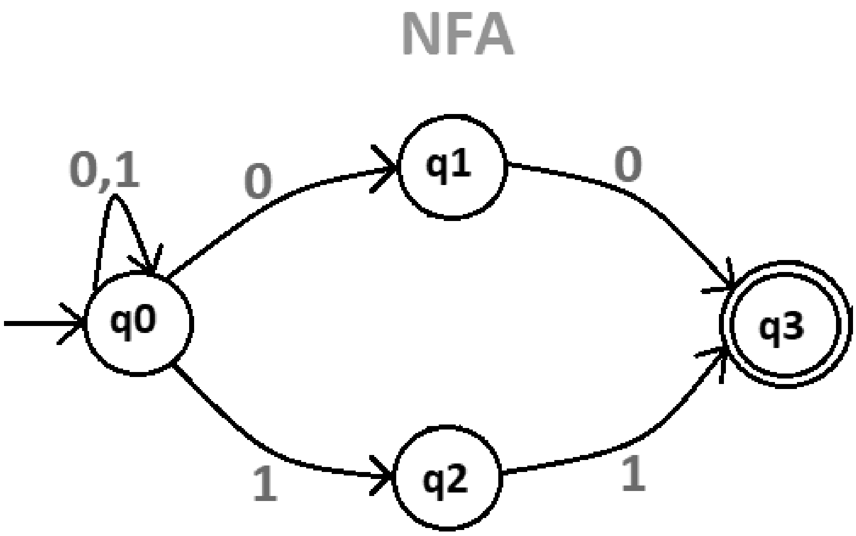


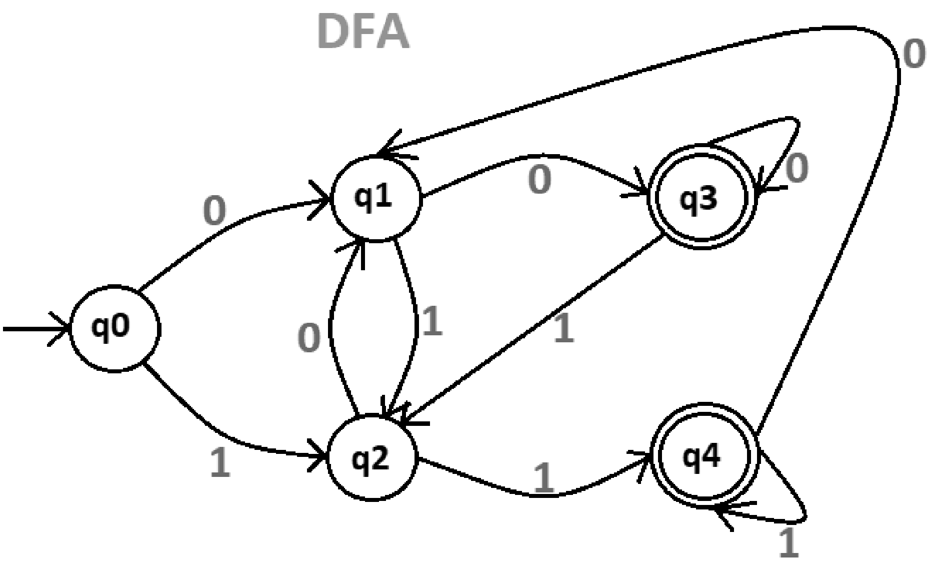
The above image is an NFA because in state q0 with input 0, we can either go to state q0 or q1.

Example 6:

Draw a deterministic and non-deterministic finite automata that accepts 00 and 11 at the end of a string containing 0, 1 in it, e.g., 01010100 but not 000111010.

Explanation – Design a DFA and NFA of the same string. If the input value reaches the final state, then it is acceptable; otherwise, it is not acceptable. The NFA of the given string is as follows:





Here, **q0** shows the initial state, **q1** and **q2** are the transition states, and **q3** and **q4** are the final states.

Note – NFA and DFA both have the same power. That means if NFA can recognize a language L, then DFA can also be defined to do so, and if DFA can recognize a language L, then NFA can also be determined to do so.

Example 7:

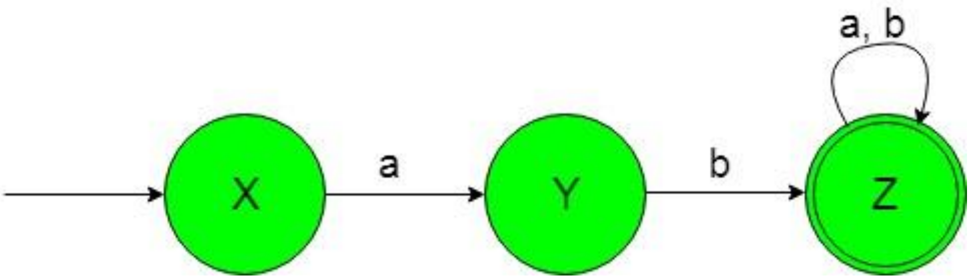
Construction of a minimal NFA accepting a set of strings over {a, b} in which each string of the language starts with ‘ab’.
Explanation: The desired language will be like:

```
L1 = {ab, abba, abaa, ...........}
```

Here as we can see that each string of the above language starts with ‘ab’ and end with any alphabet either ‘a’ or ‘b’.
But the below language is not accepted by this NFA because none of the string of below language starts with ‘ab’.

```
L2 = {ba, ba, babaaa..............}
```

The state transition diagram of the desired language will be like below:



In the above NFA, the initial state ‘X’ on getting ‘a’ as the input it transits to a state ‘Y’. The state ‘Y’ on getting ‘b’ as the input it transits to a final state ‘Z’. The final state ‘Z’ on getting either ‘a’ or ‘b’ as the input it remains in the state of itself.

Example 8:

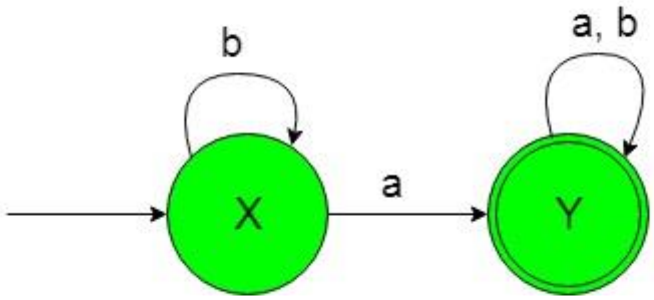
Construction of a minimal NFA accepting a set of strings over {a, b} in which each string of the language contain ‘a’ as the substring. **Explanation:** The desired language will be like:

```
L1 = {ab, abba, abaa, ...........}
```

Here as we can see that each string of the above language contains ‘a’ as the substring. But the below language is not accepted by this NFA because some of the string of below language does not contain ‘a’ as the substring.

```
L2 = {bb, b, bbbb, .............}
```

The state transition diagram of the desired language will be like below:



In the above NFA, the initial state 'X' on getting 'a' as the input it transits to a final state 'Y' and on getting 'b' as the input it remains in the state of itself. The final state 'Y' on getting either 'a' or 'b' as the input it remains in the state of itself.

Example 9:

Construction of a minimal NFA accepting a set of strings over {a, b} in which each string of the language starts with 'a'.

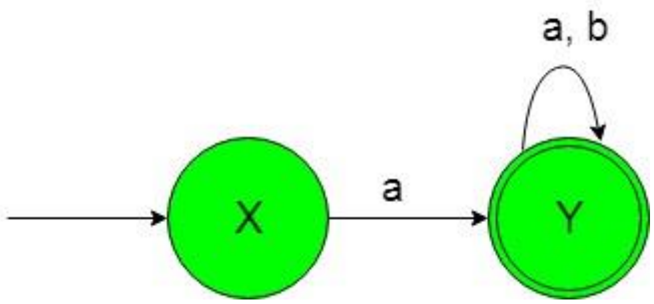
Explanation:
The desired language will be like:

```
L1 = {ab, abba, abaa, ...........}
```

Here as we can see that each string of the above language starts with 'a' and end with any alphabet either 'a' or 'b'. But the below language is not accepted by this NFA because none of the string of below language starts with 'a'.

```
L2 = {ba, ba, babaaa..............}
```

The state transition diagram of the desired language will be like below:



Example 10:

Construction of a minimal NFA accepting a set of strings over {a, b} in which each string of the language contains 'ab' as the substring.

Explanation:
The desired language will be like:

```
L1 = {ab, abba, abaa, ...........}
```

Here as we can see that each string of the above language contains 'ab' as the substring but the below language is not accepted by this NFA because some of the string of below language does not contain 'ab' as the substring.

```
L2 = {bb, b, bbbb, .............}
```

The state transition diagram of the desired language will be like below: