Create a new state for each λ-transition. In the image, the NDFA-λ has two λ-transitions: from state S to state P, and from state Q to state R. This means that we need to create two new states, P' and R', for the DFA.

Add arrows from the original state to the new state for each λ-transition. In the image, the NDFA-λ has two λ-transitions: from state S to state P, and from state Q to state R. This means that we need to add two arrows from state S to state P', and from state Q to state R', in the DFA.

Remove the λ-transitions from the original state. In the image, the NDFA-λ has two λ-transitions: from state S to state P, and from state Q to state R. This means that we need to remove these two λ-transitions from the original state in the DFA.

Create a new final state for the DFA. The DFA will have one final state, which will be the state that is reached after following all of the λ-transitions. In the image, the final state of the NDFA-λ is state R. This means that the final state of the DFA will be state R'. Here is the transition table of the equivalent DFA:

State	a	b
S	S'	P'
P'	P'	P'
Q'	Q'	R'
R'	R'	R'

You never compute the lambda=closures or the input transitions. I cannot see any justification for the procedure that you do follow. Your new state P' has two rules P' --a--> P' and P' --b--> P'. So once in state P', the machine can never move to a different state, and P' is a not final state, any string starting with b is not accepted. But S' (I assume your S is S') is also recursive and not-final, so your machine does not accept any strings at all.

The initial state of the DFA is state S, and the final state is state R'. The DFA accepts the same set of strings as the NDFA-λ. Here is a justification for each of the steps:

Step 1: Creating a new state for each λ-transition is necessary to ensure that the DFA is deterministic. If we did not create a new state for each λ-transition, then the DFA would have multiple paths for each string.

Step 2: Adding arrows from the original state to the new state for each λ-transition is necessary to ensure that the DFA accepts the same set of strings as the NDFA-λ. If we did not add these arrows, then the DFA would not be able to accept strings that start with a λ-transition.

Step 3: Removing the λ-transitions from the original state is necessary to ensure that the DFA is deterministic. If we did not remove these transitions, then the DFA would have multiple paths for each string that starts with a λ-transition.

Step 4: Creating a new final state for the DFA is necessary to ensure that the DFA accepts the same set of strings as the NDFA-λ. If we did not create a new final state, then the DFA would not be able to accept strings that end with a λ-transition.