Problem Statement: In an application, the user can take any of the following actions:
(a) define parameter 1
(b) define parameter 2
(c) calculate a value which depends on these two parameters.

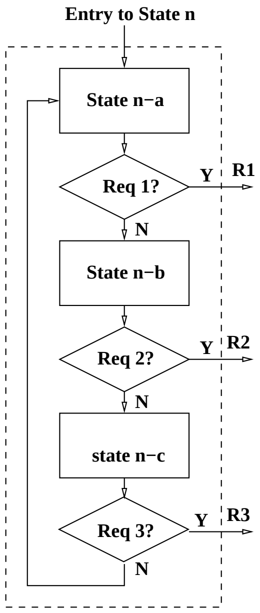
The parameters can be defined in any order. If the user asks for a calculation without defining both parameters, an error message is issued. If both parameters were defined, asking for a calculation prints the results and makes both parameters undefined. Defining a parameter again before asking for a calculation overwrites the previous value of the parameter with the new one.

Construct a state diagram for this application, assuming user actions (a), (b) or (c) to be the external inputs. The state diagram should use tests and actions, which conform to the following restrictions:
Each state is associated with a unique test.
A test can return only a yes or no answer.
Associated with either answer is an action and a next state

Give only the state diagram and the associated data structure. No program needs to be written.
Solution
Only three external events are possible in any state:
Req-1 corresponding to user wanting to set parameter 1,
Req-2 corresponding to user wanting to set parameter 2
and Req-3 corresponding to user demanding a calculation.

Any of these events can occur at any time. Therefore, each state must respond to these events. Since we are required to have just one test per state, we need to split each state of the system in 3 sub-states.

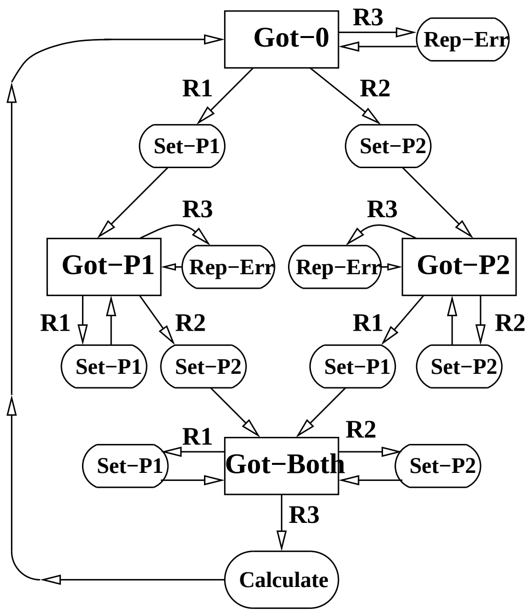
The sub-state 'a' asks "Does the user want to set parameter 1?". If the answer is yes, action corresponding to a is taken and the system goes to sub-state 'a' of the specified next state. If the answer is no, there is no action and the system goes to the sub-state 'b' of the current state.
The sub-state 'b' asks "Does the user want to set parameter 2?". If the answer is yes, action as described for this is taken and the system goes to sub-state 'a' of the specified next state. If the answer is no, there is no action and the system goes to the sub-state 'c'.
The sub-state 'c' asks "Does the user want to perform a calculation?". If the answer is yes, action as described for this request is taken and the system goes to sub-state 'a' of the specified next state. If the answer is no, there is no action and the system returns to the sub-state 'a' of the current state.



For avoiding clutter, we understand each state to be split as above and represent the group of split states a, b and c by a single rectangle with three exits marked R1, R2 and R3.

The system can be in any one of four states:

1. Got-0 : No parameters defined.
2. Got-P1: Only parameter 1 has been set.
3. Got-P2: Only parameter 2 has been set.
4. Got-Both: Both parameters have been set.



Here rectangles are triplets of states as described above, R's are exits from state splitting corresponding to Req1, Req2 and Req3 respectively, as described before. Ovals are actions taken by the system. We shall refer to each triplet state simply as "states" in the description below:

- If the system is is in Got-0 state
  Req-1 results in setting parameter 1 and the system goes to Got-P1.
  Req-2 results in setting parameter 2 and the system goes to Got-P2.
  Req-3 results in the generation of an error message and the system returns to Got-0.
- If the system is is in Got-P1 state
  Req-1 results in setting parameter 1 again and the system returns to Got-P1.
  Req-2 results in setting parameter 2 and the system goes to Got-Both.
  Req-3 results in the generation of an error message and the system returns to Got-P1.

- If the system is is in Got-P2 state
  Req-1 results in setting parameter 1 and the system goes to Got-Both.
  Req-2 results in setting parameter 2 again and the system returns to Got-P2.
  Req-3 results in the generation of an error message and the system returns to Got-P2.

- If the system is is in Got-Both state
  Req-1 results in setting parameter 1 again and the system returns to Got-Both.
  Req-2 results in setting parameter 2 again and the system returns to Got-Both.
  Req-3 results in the calculation and the system returns to Got-0.

Thus we have the following tests:
0: Does the user want to set parameter 1?
1: Does the user want to set parameter 2?
2: Does the user want to perform a calculation?

The system has the following actions:
0: Do nothing
1: Set Parameter 1
2: Set Parameter 2
3: Calculate and unset both parameters.
4: Report Error

Let us label our states as follows:

0: Got-0-a	1: Got-0-b	2: Got-0-c
3: Got-P1-a	4: Got-P1-b	5: Got-P1-c
6: Got-P2-a	7: Got-P2-b	8: Got-P2-c
9: Got-Both-a	10: Got-Both-b	11: Got-Both-c

This leads to the following state table:

Current St.	Test	Yes Case		No Case	
		Action	Next St.	Action	Next St.
0	0	1	3	0	1
1	1	2	6	0	2
2	2	4	0	0	0
3	0	1	4	0	4
4	1	2	9	0	5
5	2	4	3	0	3
6	0	1	9	0	7
7	1	2	6	0	8
8	2	4	6	0	6
9	0	1	10	0	10
10	1	2	11	0	11
11	2	3	0	0	9

This table is better understood if we replace numbers with names of tests, actions and states.

Current St.	Test	Yes Case		No Case	
		Action	Next St.	Action	Next St.
Got-0-a	Req-1?	Set-P1	Got-P1-a	Nothing	Got-0-b
Got-0-b	Req-2?	Set-P2	Got-P2-a	Nothing	Got-0-c
Got-0-c	Req-3?	Rep-Err	Got-0-a	Nothing	Got-0-a
Got-P1-a	Req-1?	Set-P1	Got-P1-b	Nothing	Got-P1-b
Got-P1-b	Req-2?	Set-P2	Got-Both-a	Nothing	Got-P1-c
Got-P1-c	Req-3?	Rep-Err	Got-P1-a	Nothing	Got-P1-a
Got-P2-a	Req-1?	Set-P1	Got-Both-a	Nothing	Got-P2-b
Got-P2-b	Req-2?	Set-P2	Got-P2-a	Nothing	Got-P2-c
Got-P2-c	Req-3?	Rep-Err	Got-P2-a	Nothing	Got-P2-a
Got-Both-a	Req-1?	Set-P1	Got-Both-c	Nothing	Got-Both-b
Got-Both-b	Req-2?	Set-P2	Got-Both-c	Nothing	Got-Both-c
Got-Both-c	Req-3?	Calc	Got-0-a	Nothing	Got-Both-a

Data structure:
We can represent the test list as an array of pointers, each pointing to the subroutine performing the test.
The action list is also an array of pointers, each pointing to the subroutine performing the actions.
The state table can be an array of structures, each structure contains:

```
int Test_index;
int Y_action_index;
int Y_next_st;
int N_action_index;
int N_next_st;
```

To conserve RAM, the structure members could be implemented as bit fields within the same word.