Problem 1 [25 pts]
Given the following StateChart, draw the equivalent Finite State Machine (FSM):
Problem 2 [25 pts]

Consider this description of a single node in a distributed sensing network:

1. At startup, the node waits to receive configuration parameters, like the required sampling period and sensor trigger threshold. It waits until a configuration message is received ("Conf").
2. The node will start sampling from its attached sensors.
3. After sample collection, the data is processed.
4. After processing, the node checks for any anomalies detected on board or from neighboring sensors (collectively represented as an "Outlier" signal).
5. If an outlier is detected, the sensor emits an alert ("Alert"), then immediately goes back to sampling.
6. If no outliers are detected, the sensor emits a confirmation ("Typ"). It then calculates any remaining time until the next sampling period and sets a timer for that length "T". The node will sleep until the timer goes off, then return to sampling (step 2).
7. At any point during steps 2-6, a low-battery ("Low") signal may be received. This will send the node into low power operating mode.
8. In low power mode, the node immediately goes to sleep.
9. In low power mode, sleep will only be woken up by a hardware message from the sensors if their readings pass a certain threshold ("Thres").
10. When the node wakes in low-power mode, it takes a sample from the sensors and goes back to sleep without processing the raw data.
11. At any point in low-power mode, if a charging cable is supplied to the node (indicated by a signal "Chrg"), it will return to normal operations and take a new sample (step 2).

a. Draw a StateChart that accurately represents the functionality of a single node. You should use superstates where appropriate to make your state diagram more readable.

b. Now assume that the node simply has one normal operation mode (no low power mode). Draw the equivalent SDL representation. Apart from the low power mode, your answer must be consistent with the StateChart shown in part a).
Problem 3 [25]
Consider this SDF graph and answer the following short questions:

a. Each time node A executes, it produces / consumes _____ token(s).
b. Each time node B executes, it produces / consumes _____ token(s).
c. There is/are ______ token(s) stored on the arc at start up.
d. The sample produced at node A’s 1st invocation is consumed at node B’s ___st/nd/th invocation.

Consider another SDF graph:

e. Write down the incidence matrix. Use columns to represent nodes and rows to represent edges.
f. What is the rank of the matrix?
g. How can we infer the existence of a periodic admissible sequential schedule (PASS) based on this rank?
h. Show the number of invocations for each node in 1 cycle of the schedule. (E.g. The schedule "AABC" would give answers A=2, B=1, C=1).
   A:
   B:
   C:
   D:
i. Find the initial number of elements in each edge buffer for your PASS schedule (Hint: execute one cycle of your PASS schedule manually to find the number of initial elements necessary in each buffer.)
j. Find the minimum size required of each edge buffer.
**Problem 4 [25 pts]**

Examine the following Esterel code snippet:

```esterel
await A emit M;
present B then
  emit N
end present;
pause;
abort
  loop
  present C then emit O else emit P end;
pause;
end loop
when D;
emit Q;
```

a) Given the following timeline and input sequence, list all expected outputs at each time point.  
*Hint: You may find this extensive reference paper useful:*

[https://www.researchgate.net/publication/242374294_The_Esterel_v5_Language_Primer_Version_v5_91](https://www.researchgate.net/publication/242374294_The_Esterel_v5_Language_Primer_Version_v5_91)

<table>
<thead>
<tr>
<th>input</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*b) Draw the FSM (finite state machine) that accurately presents functionality of the Esterel code.  
*Hint: be mindful of the differences between instantaneous statements and delayed statements.*

 output