**CS 3-5274: Modeling & Simulation**

**Assignment 1**

A.

* To run the Moore and Mealy machines, simply open the files “HW1\_Moore” and “HW1\_Mealy” respectively and click the green “Run” button in the Toolbars.
* Click the buttons (e.g. Soda, Coffee, Snack, Nickel, Dime, Quarter, and Cancel) in the Simulink model as you desire and see the displays (1, 2, and 3) as you expected.  
  **NB:** Click the buttons using the mouse: **press the mouse left click for at least 1/2 a second otherwise the model cannot read the input event if the button is very quickly pressed**.

B.

1. Mealy FSM
   1. Inputs Events: Soda, Coffee, Snack, Nickel, Dime, Quarter, and Cancel

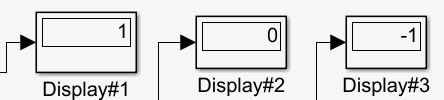
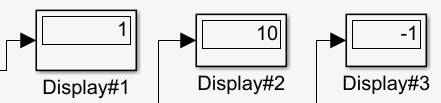
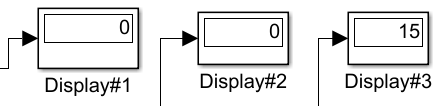
Output Data: Display1 (Product Chosen), Display2 (Amount Put in), Display3 (Change\Return)

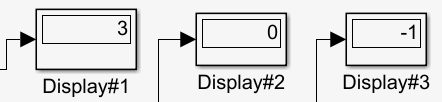
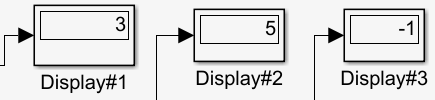
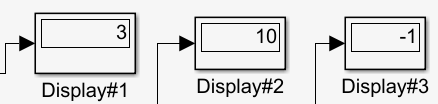
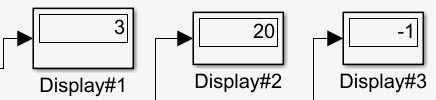
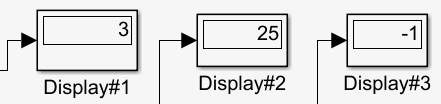
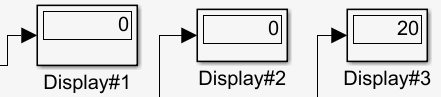
States: Wait (Idle), Soda, SodaCycle, Coffee, CoffeeCycle Snack, SnackCycle,

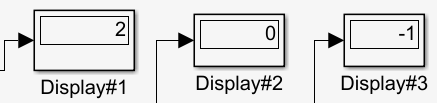
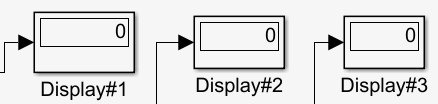
Release/Reset, Five cents, Ten cents, Fifteen cents, Twenty cents, TwentyFive cents, Thirty Cents

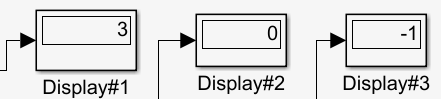
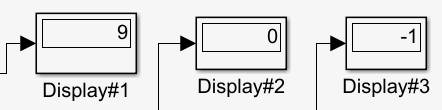
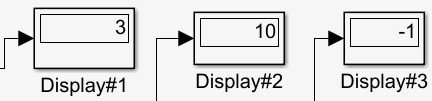
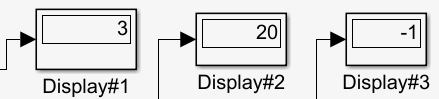
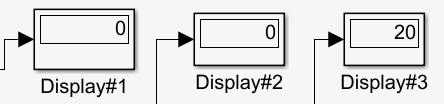
Transitions: Solely based on Input Events- Soda, Coffee, Snack, Nickel, Dime, Quarter, and Cancel. There is also a local event Reset generated after completing a transaction.

* 1. 4 example Runs

Soda, Dime, Quarter  
  
  


Snack, Nickel, Nickel, Dime, Nickel, Quarter  
  
  
  
  
  


Coffee, Quarter  
  


Snack, Coffee, Dime, Dime, Cancel (Failed Transaction)  
  
  
  
  


* 1. The implementation I have made is very robust to all the types of input. Instead of using a selector switch to generate the event, I have used buttons to generate input events to the Stateflow machine. This makes it more realistic.
  2. There is only **one** issue in the model. The buttons can sometimes not properly generate an input event if they are clicked very quickly.
  3. To generate the model, I conceptualized what are the possible set of inputs, outputs and states. The inputs and outputs were pretty straightforward to determine. However, determining the states was crucial to making the model more efficient and preventing any form of state space explosion. So I decided that the model should have states identical to a real vending machine. These are usually given by the type of food ordered permuted with the type of coins put into the machine. That’s how I formulated the FSM.  
       
     The FSM modeling is a good representation to generate the solution to the problem. However since cost can vary in many ways, there can be problems if large permutations of coins are used causing state space explosion.   
       
     Here efficiency is achieved due to the lesser number of states being used by the Mealy Machine compared to the Moore Machine.

1. Moore FSM
   1. Inputs Events: Soda, Coffee, Snack, Nickel, Dime, Quarter, and Cancel

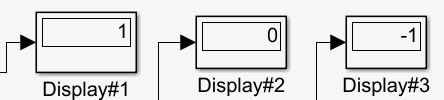
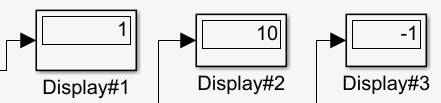
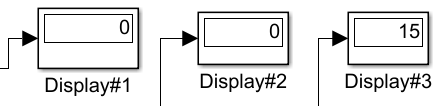
Output Data: Display1 (Product Chosen), Display2 (Amount Put in), Display3 (Change\Return)

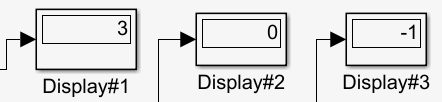
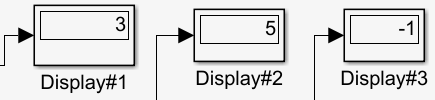
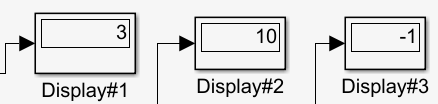
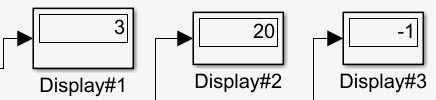
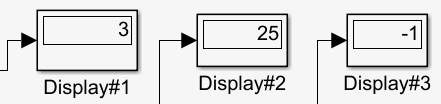
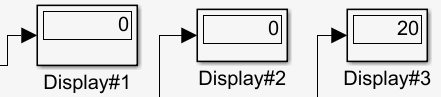
States: Wait (Idle), Soda, SodaCycle, Coffee, CoffeeCycle Snack, SnackCycle,

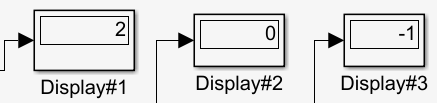
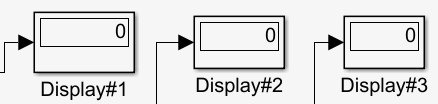
Release/Reset, Five cents, Ten cents, Fifteen cents, Twenty cents, TwentyFive cents, Thirty Cents, Err1(Error State), {S1,S2,S3,S4,S5}(Substates within each coin state.)

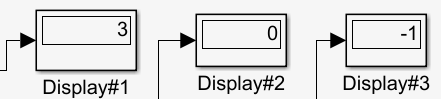
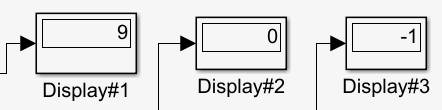
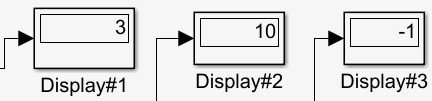
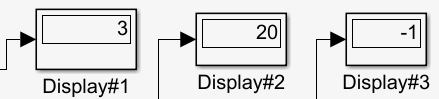
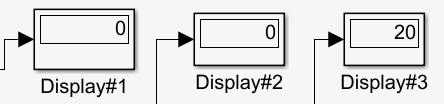
Transitions: Solely based on Input Events- Soda, Coffee, Snack, Nickel, Dime, Quarter, and Cancel. There is also a local event Reset generated after completing a transaction.

* 1. 4 example Runs

Soda, Dime, Quarter  
  
  


Snack, Nickel, Nickel, Dime, Nickel, Quarter  
  
  
  
  
  


Coffee, Quarter  
  


Snack, Coffee, Dime, Dime, Cancel (Failed Transaction)  
  
  
  
  


* 1. The implementation I have made is very robust to all the types of input. Instead of using a selector switch to generate the event, I have used buttons to generate input events to the Stateflow machine. This makes it more realistic.
  2. There is only **one** issue in the model. The buttons can sometimes not properly generate an input event if they are clicked very quickly.
  3. To generate the model, I conceptualized what are the possible set of inputs, outputs and states. The inputs and outputs were pretty straightforward to determine. However, determining the states was crucial to making the model more efficient and preventing any form of state space explosion. So I decided that the model should have states identical to a real vending machine. These are usually given by the type of food ordered permuted with the type of coins put into the machine. That’s how I formulated the FSM.  
       
     The FSM modeling is a good representation to generate the solution to the problem. However since cost can vary in many ways, there can be problems if large permutations of coins are used causing state space explosion.

Here efficiency is less due to the more number of states being used by the Moore Machine compared to the Mealy Machine.

III

***Extra Credit***

* I reduced the number of transitions by a significant amount by using **concurrent** programming
* In the Moore machine, I did not create a separate state for the cancel but **reutilized** the final states using transitions from coin state to the final state.