	QUARTER	SLECET	CANCEL	RETRIEVE
IDLE	SELECT | NONE	IDLE | NONE	IDLE | QUARTER	IDLE | NONE
SELECT	SELECT | NONE	VENDED | GUM or GRANOLA or NONE	IDLE | QUARTER	SELECT | NONE
VENDED	VENDED | QUARTER	VENDED | NONE	VENDED | NONE	IDLE | QUARTER

In my original implementation there was some confusion on the behavior while in the VEND state and a bug in the implementation of the insert quarter logic which made the machine dispense change while in VEND. I started addressing these issues by first redefining my VEND state as VENDED (these updates can be seen in the transition table above), to more accurately represent the state I am trying to model. VENDED is where the item has already been dispensed and the machine is waiting for the output receptacle to be emptied so it can return to idle. When in this state you can insert more quarters if you want but they will not be added to the balance but instead returned to the customer via being added to the change return receptacle. (Change return does not mean how much money you got back after the transaction is done but the receptacle on a vending machine that holds all the coins that come out when you get coins back) Hence why the change return count increases every time you try to insert a quarter while it is in the VENDED state. Additionally, the CANCEL input no longer returns your change because the transaction is already complete and there is nothing to cancel anymore. The SELECT input also will return nothing and not

change the state while in VENDED. Finally, the RETRIEVE action empties the output receptacle of whatever item was purchased in the SELECT state and outputs the change you are owed into the change return. This is because the transaction is now fully complete and you can only buy one item per transaction, hence why you need to claim your item before inserting money and being able to buy another item. I believe this thoroughly explains the issue of why the change keeps going up when you insert QUARTERS while in the VENDED state. Additionally, the take change button is not an input for the FSM because it has no real effect on how the vending machine operates, it is just an extra feature I added because it adds to the experience.

As for the bug causing the machine to return all of the money in the input balance when you inserted a QUARTER for the first time after entering the VEND state. I fixed it by implementing some conditionals in the UpdateChange() method so that it would have specific logic just for when the state is VENDED, and the change count is updated.

Now for what I learned by updating this assignment, my biggest take away is that the language you choose when defining your states and the transition table is critical for others understanding how the FSM will work. Changing the VEND

state to VENDED better illustrates that it is not currently vending the item, dispensing it, but in a state after the item being dispensed to the customer where it thus makes sense for the behaviors that I designed it to follow.