Dispense15 with Change, of the given figure is a modification of the previous dispenser, but this version also outputs change. For example, when in state 10, and an dime is input, an item is output and change (of 5 cents) is output also, and the next state is 0.

Tables of transitions also describe such sequential systems, as shown below to the right. For each of the 3 states there are 2 possible coins input thus yielding 6 possible combinations. Such representations help check that all possibilities are considered.

<table>
<thead>
<tr>
<th>state now</th>
<th>coin input</th>
<th>state next</th>
<th>actions output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>nickel</td>
<td>5</td>
<td>none</td>
</tr>
<tr>
<td>0</td>
<td>dime</td>
<td>10</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>nickel</td>
<td>10</td>
<td>none</td>
</tr>
<tr>
<td>5</td>
<td>dime</td>
<td>0</td>
<td>item</td>
</tr>
<tr>
<td>10</td>
<td>nickel</td>
<td>0</td>
<td>item</td>
</tr>
<tr>
<td>10</td>
<td>dime</td>
<td>0</td>
<td>item &amp; 5</td>
</tr>
</tbody>
</table>

Programs corresponding to such state diagrams are easy to do, as follows. They consist of a large choice, corresponding to the states, and then for each state is a sub choice depending on the input. Notice that the repeat on false causes the loop to continue forever.

```plaintext
Boxes state, coin ofType int
Set state = 0
Repeat
ExitOn (false) -- loop forever!
Output "Enter coin " -- prompt
Input coin -- enter
Outputln coin -- echo

If (state == 0) then
    If (coin == 5) then
        Set state = 5
    Else -- coin is 10
        Outputln "Give item "
        Set state = 0
    EndIf -- state is 5
Else -- (state is 10)
    If (coin == 5) then
        Outputln "Give item "
        Set state = 0
    Else -- coin is 10
        Outputln "Give item "
        Outputln "Give change "
        Set state = 0
    EndIf -- state is 10
EndIf
EndRepeat
```