```
1  using System;
2  using System.Collections.Generic;
3
4  namespace QLearningEX
5  {
6      class Program
7      {
8          static string[] TriggerTrue = new string[]{
9              "---",
10             "| |",
11             "| |",
12             "MMM",
13             "WWW",
14             "   ",
15             "   ",
16             "   "
17         };
18         static string[] TriggerFalse = new string[]{
19             "   ",
20             "   ",
21             "   ",
22             "MMM",
23             "WWW",
24             "| |",
25             "| |",
26             "---"
27         };
28
29         static string[] statesString = new string[]
30             {// 0     1     2     3     4     5     6     7
31                 "000","100","010","001","110","101","011","111"
32             };
33
34         static bool[] Triggers = new bool[] { false, false, false };
35         static double GAMMA = 0.8;
36         static double ALPHA = 0.1;
37         static string agentState;
38
39         static Dictionary<string, double[]> Q = new Dictionary<string, double[]>();
40         static Dictionary<string, double[]> R = new Dictionary<string, double[]>();
41
42         static void Main(string[] args)
43         {
44             SetupQ();
45             WriteTriggers();
46             WaitForInput();
47
48             Learn();
49             PrintSequences();
50
51             WaitForInput();
52         }
53         static void SetupQ()
54         {
55
56
57             for (int state = 0; state < statesString.Length; state++)
58             {
59                 Q[statesString[state]] = new double[3];
60                 R[statesString[state]] = new double[3];
61                 for (int action = 0; action < 3; action++)
62                 {
63
64                     Q[statesString[state]][action] = 0;
```

```
65                      if ((state == 4 && action == 2)
66                          || (state == 5 && action == 1)
67                          || (state == 6 && action == 0))
68                      {
69                          R[statesString[state]][action] = 100;
70                      }
71                      else
72                      {
73                          R[statesString[state]][action] = 0;
74                      }
75
76                  }
77              }
78          }
79
80          static void Learn()
81          {
82              WriteRMatrix();
83              WriteQMatrix();
84              Random rand = new Random();
85
86              int nextAction = 0;
87              string nextState = "";
88              int tr = 0;
89              double hV = 0;
90
91              double v, r;
92
93              while (tr < 500)
94              {
95                  agentState = statesString[rand.Next(0, 7)];
96                  SetTriggersToString(agentState);
97
98                  while (true)
99                  {
100                     nextAction = rand.Next(0, 3);
101                     nextState = ChangeTrigger(nextAction);
102                     Console.WriteLine(agentState + "\t" + nextAction + "\nNextState: " +
                          nextState);
103
104                     hV = FindHighestVal(Q[nextState]);
105                     r = R[agentState][nextAction];
106                     double newQ = r + GAMMA * hV;
107                     double oldQ = Q[agentState][nextAction];
108                     Q[agentState][nextAction] = oldQ + ALPHA * (newQ - oldQ);
109
110                     WriteQMatrix();
111                     agentState = TriggersToString();
112                     if (agentState == "111")
113                         break;
114                 }
115                 Console.WriteLine();
116                 Console.WriteLine("----------------------End Of
                      Test----------------------");
117                 Console.WriteLine();
118                 tr++;
119             }
120             WaitForInput();
121
122         }
123
124         static void PrintSequences()
125         {
126             int nextAction;
```

```
127             for (int initial = 0; initial < 7; initial++)
128             {
129                 agentState = statesString[initial];
130                 Console.WriteLine("-----------------------------------------------------");
131                 Console.WriteLine("-----------------------------------------------------");
132                 Console.Write("\n" + initial + ": " + agentState + "\n");
133 
134 
135                 SetTriggersToString(agentState);
136                 while (true)
137                 {
138                     Console.Write("\n" + agentState);
139                     Console.Write(" - ");
140                     nextAction = FindHighestIndex(Q[agentState]);
141                     for (int a = 0; a < Q[agentState].Length; a++)
142                         Console.Write(Q[agentState][a] + "   ");
143 
144                     Console.Write(" :  " + nextAction);
145                     Console.Write("\n");
146 
147                     agentState = ChangeTrigger(nextAction);
148                     if (agentState == "111")
149                         break;
150 
151                 }
152                 Console.Write("\n" + agentState + "\n");
153             }
154         }
155         static int FindHighestIndex(double[] vals)
156         {
157             int index = 0;
158             double highest = 0;
159 
160             for (int a = 0; a < vals.Length; a++)
161             {
162 
163                 if (vals[a] > highest)
164                 {
165                     highest = vals[a];
166                     index = a;
167                 }
168             }
169 
170             return index;
171         }
172         static double FindHighestVal(double[] vals)
173         {
174             double hVal = -1;
175 
176             for (int a = 0; a < vals.Length; a++)
177             {
178                 if (vals[a] > hVal)
179                 {
180                     hVal = vals[a];
181                 }
182             }
183 
184             return hVal;
185         }
186 
187         static string ChangeTrigger(int trigger)
188         {
189             Triggers[trigger] = !Triggers[trigger];
190 
```

```
191             return TriggersToString();
192         }
193         static void SetTriggersToString(string input)
194         {
195             for (int a = 0; a < Triggers.Length; a++)
196             {
197                 Triggers[a] = input[a] == '0' ? false : true;
198 
199             }
200         }
201 
202         static void WriteQMatrix()
203         {
204             Console.WriteLine();
205             Console.WriteLine("Q");
206             Console.WriteLine();
207             for (int state = 0; state < 8; state++)
208             {
209                 Console.Write(statesString[state] + ":  ");
210                 for (int action = 0; action < 3; action++)
211                 {
212                     int value = (int)Q[statesString[state]][action] == 100 ? 99 : (int)Q
                        [statesString[state]][action];
213                     Console.Write(value + "     \t");
214                 }
215                 Console.Write("\n");
216             }
217         }
218         static void WriteRMatrix()
219         {
220             Console.WriteLine();
221             Console.WriteLine("R");
222             Console.WriteLine();
223             for (int state = 0; state < 8; state++)
224             {
225                 for (int action = 0; action < 3; action++)
226                 {
227                     Console.Write(R[statesString[state]][action] + "\t");
228                 }
229                 Console.Write("\n");
230             }
231         }
232 
233         static void WaitForInput()
234         {
235             Console.Read();
236             Console.Read();
237 
238         }
239         static string TriggersToString()
240         {
241             string s = "";
242             for (int a = 0; a < Triggers.Length; a++)
243             {
244                 s += Triggers[a] == true ? 1 : 0;
245             }
246             return s;
247         }
248 
249         public static void WriteTriggers()
250         {
251 
252 
253             for (int row = 0; row < 8; row++)
```

```
254                 {
255                     if (row == 3)
256                     {
257                         Console.Write("___");
258                     }
259                     else
260                     {
261                         Console.Write("   ");
262                     }
263                     for (int a = 0; a < Triggers.Length; a++)
264                     {
265                         if (Triggers[a] == true)
266                         {
267                             Console.Write(TriggerTrue[row]);
268                         }
269                         else
270                         {
271                             Console.Write(TriggerFalse[row]);
272                         }
273                         if (row == 3)
274                         {
275                             Console.Write("___");
276                         }
277                         else
278                         {
279                             Console.Write("   ");
280                         }
281                     }
282                     Console.Write("\n");
283                 }
284 
285             }
286         }
287     }
288 
```