```
 1: #include <iostream>
 2: #include <string>
 3: using namespace std;
 4:
 5: class Television {
 6:
 7:   /* class-level attributes */
 8:   static const int MIN_VOLUME = 0;
 9:   static const int MAX_VOLUME = 10;
10:   static const int MIN_CHANNEL = 2;
11:   static const int MAX_CHANNEL = 99;
12:
13:  private:
14:   // Data members of instance
15:
16:   /** Whether the power is on */
17:   bool    powerOn;
18:
19:   /** Whether the tv is muted */
20:   bool    muted;
21:
22:   /** The current volume level */
23:   int     volume;
24:
25:   /** The most recent previous channel number */
26:   int     prevChan;
27:
28:  public:
29:
30:   /** Creates a new Television instance.
31:    *
32:    *  The power is initially off.  Upon the first time the TV is turned on,
33:    *  it will be set to channel 2, and a volume level of 5.
34:    */
35:   Television()  {
36:     powerOn = false;
37:     muted = false;
38:     volume = 5;
39:     channel = 2;
40:     prevChan = 2;
41:   }
42:
43:   /** Toggles the power setting.
44:    *
45:    *  If Television is off, turns it on.
46:    *  If Television is on, turns it off.
47:    */
48:   togglePower() { powerOn = !powerOn; }
49:
50:   /** Toggles the setting for mute.
51:    *
52:    *  If power is off, there is no effect.
53:    *
54:    *  Otherwise, if television was unmuted, it becomes muted.
55:    *  If television was muted, it becomes unmuted and the volume is
56:    *  restored to its previous setting.
57:    */
58:   void toggleMute()  {
59:     if (powerOn)
60:       muted = !muted;
61:   }
62:
63:   /** Increments the volume of the Television by one increment.
64:    *
65:    *  If power is currently off, there is no effect (-1 returned).
```

```
 66:    *  Otherwise, updates the volume setting appropriately.
 67:    *
 68:    *  If volume was at maximum level, it remains at maximum level.
 69:    *  If television is currently muted, it will be unmuted as a result.
 70:    *
 71:    *  @return  the resulting volume level
 72:    */
 73:   int volumeUp()  {
 74:     if (powerOn) {
 75:       if (volume < MAX_VOLUME)
 76:         volume++;
 77:       muted = false;
 78:       return volume;
 79:     } else
 80:       return -1;
 81:   }
 82:
 83:   /** Decrements the volume of the Television by one increment.
 84:    *
 85:    *  If power is currently off, there is no effect (-1 returned).
 86:    *  Otherwise, updates the volume setting appropriately.
 87:    *
 88:    *  If volume was at minimum level, it remains at minimum level.
 89:    *  If television is currently muted, it will be unmuted as a result.
 90:    *
 91:    *  @return  the resulting volume level
 92:    */
 93:   int volumeDown()  {
 94:     if (powerOn) {
 95:       if (volume > MIN_VOLUME)
 96:         volume--;
 97:       muted = false;
 98:       return volume;
 99:     } else
100:       return -1;
101:   }
102:
103:   /** Increments the channel.
104:    *
105:    *  If power is off, there is no effect (-1 returned).
106:    *  Otherwise, updates the channel setting appropriately.
107:    *
108:    *  If channel had been set to the maximum of the valid range of
109:    *  channels, the effect will be to 'wrap' around resulting in the
110:    *  channel being set to the minimum channel.
111:    *
112:    *  @return The resulting channel setting
113:    */
114:   int channelUp()  {
115:     if (powerOn) {
116:       prevChan = channel;
117:       channel++;
118:       if (channel > MAX_CHANNEL)
119:         channel = MIN_CHANNEL;    // wrap around
120:       return channel;
121:     } else
122:       return -1;
123:   }
124:
125:   /** Decrements the channel.
126:    *
127:    *  If power is off, there is no effect (-1 returned).
128:    *  Otherwise, updates the channel setting appropriately.
129:    *
130:    *  If channel had been set to the minimum of the valid range of
```

```
131:    *  channels, the effect will be to 'wrap' around resulting in the
132:    *  channel being set to the maximum channel.
133:    *
134:    *  @return The resulting channel setting
135:    */
136:   int channelDown()  {
137:     if powerOn {
138:       prevChan = channel;
139:       channel--;
140:       if (channel < MIN_CHANNEL)
141:         channel = MAX_CHANNEL;     // wrap around
142:       return channel;
143:     } else
144:       return -1;
145:   }
146:
147:   /** Sets the channel to given number (if valid).
148:    *
149:    *  If power is off, there is no effect.
150:    *  If given number is illegal channel, no effect.
151:    *
152:    *  @param  number   the desired channel number
153:    *  @return  true if change was enacted; false otherwise.
154:    */
155:   bool setChannel(number)  {
156:     if ((powerOn) && (MIN_CHANNEL <= number) && (number <= MAX_CHANNEL)) {
157:       prevChan = channel;      // must record this before it is lost
158:       channel = number;
159:       return true;
160:     } else
161:       return false;
162:   }
163:
164:   /** Changes the channel to most recent, previously viewed.
165:    *
166:    *  If power is off, there is no effect.
167:    *
168:    *  @return  the resulting channel setting
169:    */
170:   int jumpPrevChannel() const {
171:     if (powerOn) {
172:       int temp;
173:       temp = channel;
174:       channel = prevChan;
175:       prevChan = temp;
176:       return channel;
177:     } else
178:       return -1;
179:   }
180:
181:   /* allows private access to external function */
182:   friend ostream& operator<<(ostream&, const Television&);
183: };
184:
185:
186: /*
187:  * Overloading the output operator.
188:  */
189: ostream& operator<<(ostream& out, const Television& tv) {
190:   out << "Power setting is currently       "
191:       << (tv.powerOn ? "true" : "false") << endl
192:       << "Channel setting is currently     "
193:       << tv.channel << endl
194:       << "(previous channel) is currently  "
195:       << tv.prevChan << endl
```

```
196:       << "Volume Setting is currently      "
197:       << tv.volume << endl
198:       << "Mute is currently                "
199:       << (tv.muted ? "true" : "false") << endl;
200:   return out;
201: }
202:
203: /** Sample unit test. */
204: int main() {
205:
206:   Television sony;    // uses the DEFAULT constructor
207:   cout << "Newly created television:" << endl;
208:   cout << sony << endl << endl;
209:
210:   sony.channelUp();
211:   cout << "After call to channelUp():" << endl;
212:   cout << sony << endl << endl;
213:
214:   sony.togglePower();
215:   cout << "After call to togglePower():" << endl;
216:   cout << sony << endl << endl;
217:
218:   sony.setChannel(22);
219:   cout << "After call to setChannel(22):" << endl;
220:   cout << sony << endl << endl;
221:
222:   sony.jumpPrevChannel();
223:   cout << "After call to jumpPrevChannel():" << endl;
224:   cout << sony << endl << endl;
225:
226:   sony.jumpPrevChannel();
227:   cout << "After another call to jumpPrevChannel():" << endl;
228:   cout << sony << endl << endl;
229:
230:   sony.channelUp();
231:   cout << "After call to channelUp():" << endl;
232:   cout << sony << endl << endl;
233:
234:   sony.jumpPrevChannel();
235:   cout << "After call to jumpPrevChannel():" << endl;
236:   cout << sony << endl << endl;
237:
238:   sony.toggleMute();
239:   cout << "After call to toggleMute():" << endl;
240:   cout << sony << endl << endl;
241:
242:   sony.volumeUp();
243:   cout << "After call to volumeUp():" << endl;
244:   cout << sony << endl << endl;
245:
246:   // try to max-out the volume
247:   for (int i=0; i<250; i++)
248:     sony.volumeUp();
249:   cout << "After 250 calls to volumeUp():" << endl;
250:   cout << sony << endl << endl;
251:
252:   // try to wrap-around the channel
253:   for (int i=0; i<250; i++)
254:     sony.channelDown();
255:   cout << "After 250 calls to channelDown():" << endl;
256:   cout << sony << endl << endl;
257: }
```