The Bumper Turtles model created in this lab requires the use of *Boolean logic* and *conditional control flow*. The basic rules are:

1. Each turtle starts in the middle of a patch. This could be any patch, but not part in one patch and part in another.
2. Every tick, each turtle “looks” ahead one patch along its current heading.
   a. If the patch ahead is black then the turtle makes a U-Turn.
   b. If the patch ahead is blue, then the turtle makes a 90° left turn.
   c. If the patch ahead is red, then the turtle makes a 90° right turn.
   d. If the patch ahead is beautiful green grass, then there are two cases to deal with: If there is another turtle in that patch, then the turtle makes a U-Turn, otherwise, the turtle runs one step forward in the turf.

Note: the turtle should ONLY move when the patch ahead is green. This is because after the turtle turns, there might be another block in its new forward direction.
Your turtles’ tracks can be as simple or as complex as you’d like, with the following requirements:

1. You have at least 2 turtles, which must start at the same unique locations each time.
2. There are at least 10 black, red or blue patches that affect turtle behavior.
3. There is at least one patch where two different turtle paths cross.
4. Your turtles must stay on the tracks you create. These tracks can change, but when a turtle turns and heads in a new direction, there must be another turtle or a patch that will push it back on track. Your turtles shouldn’t wrap around the world endlessly.

**World Settings**

Use the following settings for the interface:

- \( \text{min-pxcor} = -16, \quad \text{max-pxcor} = 16, \quad \text{min-pycor} = -16, \quad \text{max-pycor} = 16 \)

**Setup Button**

Your program has a setup button that when pressed must:

1. Clear the 2D world view.
2. Remove all the turtles (clear-all does both 1 and 2).
3. Set all patches to green. Then set specific, hardcoded patches to black, blue or red. You choose which specific patches to set black, blue or red so that the turtles you create in step (4) follow run around on a cool, creative, crisscrossing track. The screen capture on the first page shows an example of such a setup after the turtles have run around a bit. You may create more turtles and more interesting paths then that shown. Hint: Draw your pattern on graph paper before coding it.
4. Create at least 2 turtles each with a specific location and heading so that someplace along the path or that will enter the path created in step (3). Make sure the pen is down.

The example image has two setup buttons: setupRandom and setupCrazy. You only need one Setup button, and it should not create turtles or patches in random locations. However, if you would like to experiment with other setups, you can!

<table>
<thead>
<tr>
<th>Hint: Setting a Particular Patch to a Specified Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>The NetLogo command:</td>
</tr>
<tr>
<td>( \text{ask patch 2 -4 { set pcolor blue }} )</td>
</tr>
<tr>
<td>will set the color of the patch with coordinates (2, -4) to blue</td>
</tr>
</tbody>
</table>
Hint: Setting a Particular Turtle to a Location and Heading

Each turtle in NetLogo has a unique identification number. The identification numbers always start with 0 and count up to one less than the total number of turtles. The NetLogo who command reports a turtle’s identification number.

These facts can be used to set properties of a particular turtle. For example:

```netlogo
1) create-turtles 2
2) [ 
3)   if (who = 0) 
4)     [ 
5)     setxy -3 0 ;; Lines 5 & 6 only execute for the turtle with ID number = 0 
6)     set heading 180 
7)   ] 
8)   if (who = 1) 
9)     [ 
10)   setxy 4 0 ;; Lines 10 & 11 only execute for the turtle with ID number = 1. 
11)   set heading 0 
12) ] 
13) ]
```

Hint: Getting the Color of the Patch Ahead
See “Bumper Turtles” video for more details

We have seen before that within a turtle context, pcolor is the color of the patch the turtle is on. In this lab, we need to look at the color of the patch one ahead of the current turtle location. This can be done with the patch-ahead function as shown below:

```netlogo
ask turtles
|
let colorOfPatchAhead green
ask patch-ahead 1 ;; 1 is the number of patches ahead to look
|
set colorOfPatchAhead pcolor ;; stores color of the patch so you
| ;; can use it in Boolean expressions
|
if (colorOfPatchAhead = blue)
|
| ;; In this code block, code what you want to happen when
| ;; the color or the patch ahead is blue.
```
To determine whether there is another turtle on the patch ahead of the current turtle, we build a triple compound command.

Within a turtle context, `patch-ahead 1`, reports the patch that is one patch ahead of the turtle’s current location.

The NetLogo function, `turtles-on patch`, reports the set of all turtles that are on `patch`.

The NetLogo function, `any? agentset`, reports true if `agentset` contains at least one agent. Otherwise, it reports false.

Putting this all together, the statement:

```
  any? turtles-on patch-ahead 1
```

Reports true if and only if there is at least one turtle on the patch that is one ahead of the current turtle’s location.

Since this function reports a Boolean (true or false) it can be used in an `if` or in an `ifelse` statement.

---

### Hint: Finding Out Whether there is a Turtle Ahead

See “Bumper Turtles” video for more details

<table>
<thead>
<tr>
<th>To determine whether there is another turtle on the patch ahead of the current turtle, we build a triple compound command.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within a turtle context, <code>patch-ahead 1</code>, reports the patch that is one patch ahead of the turtle’s current location.</td>
</tr>
<tr>
<td>The NetLogo function, <code>turtles-on patch</code>, reports the set of all turtles that are on <code>patch</code>.</td>
</tr>
<tr>
<td>The NetLogo function, <code>any? agentset</code>, reports true if <code>agentset</code> contains at least one agent. Otherwise, it reports false.</td>
</tr>
<tr>
<td>Putting this all together, the statement:</td>
</tr>
<tr>
<td><code>any? turtles-on patch-ahead 1</code></td>
</tr>
<tr>
<td>Reports true if and only if there is at least one turtle on the patch that is one ahead of the current turtle’s location.</td>
</tr>
<tr>
<td>Since this function reports a Boolean (true or false) it can be used in an <code>if</code> or in an <code>ifelse</code> statement.</td>
</tr>
</tbody>
</table>