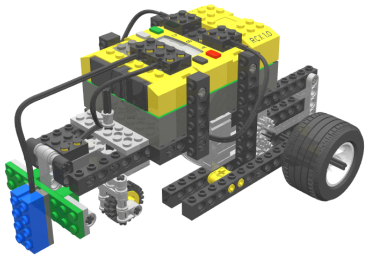
Name: _____

NQC – Using Touch Sensors

The Lego Mindstorms kit comes with two different types of sensors: touch and light. Your GT Scooter should have at least one touch sensor as part of its bumper and one light sensor in the front pointing down. If you built a dual bumper GT Scooter, you can connect both the touch sensors wires to the same input port so that it will act like a single bumper.



Using the touch sensor

When you use sensors in a NQC program, your program needs to tell the RCX what kind of sensor it is using. Here is the command for telling the RCX you have a touch sensor connected to the sensor 1 input port:

```
SetSensor(SENSOR_1, SENSOR_TOUCH);
```

One way to program a bumper is to use `until(sensor)` instead of `Wait(time)`. For example, here is the first program we talked about last time with `until` instead of `Wait`.

```
task main()
{
  SetSensor(SENSOR_1, SENSOR_TOUCH);
  On(OUT_A+OUT_C);
  until(SENSOR_1);
  Off(OUT_A+OUT_C);
}
```

Note that the **`until`** statement is not capitalized. What do you think this program will do?

Program sample	**Description**

`task main()`
`{`
`  SetSensor(SENSOR_1, SENSOR_TOUCH);`
`  until(SENSOR_1);`
`  PlaySound(SOUND_CLICK);`
`}`
	Tell RCX you are using a touch sensor on input port 1.
	Waits until touch sensor is pressed.
	Plays a click sound.

`task main()`
`{`
`  SetSensor(SENSOR_1, SENSOR_TOUCH);`
`  repeat(4) {`
`  until(SENSOR_1);`
`  PlaySound(SOUND_CLICK);`
`  Wait(20);`
`  }`
`  PlaySound(SOUND_DOWN);`
`}`
	What will this do?

`task main()`
`{`
`  SetSensor(SENSOR_1, SENSOR_TOUCH);`
`  while(true) {`
`  until(SENSOR_1);`
`  PlaySound(SOUND_CLICK);`
`  Wait(20);`
`  }`
`}`
	What about this one.

Combine what you learned last time with using `On`, `Off`, `Fwd`, `Rev`, and `Wait` commands with `SetSensor` and `until(sensor)` to make a robot that drives straight and backs up and turns when it hits things.

```
task main()
{
  SetSensor(SENSOR_1, SENSOR_TOUCH);









}
```

Multiple Tasks

Another great way to program with sensors is to use multiple tasks. So far, all the programs we have done only had one task: `task main`. But lets say you want to program your robot to drive around in a pattern and every time it runs into something, back up and turn. That's very tricky with one task but easy with two. Consider this example:

```
task DriveAround()
{
  while(true) {
    On(OUT_A+OUT_C);
    Wait(200);
    Off(OUT_A);
    Wait(100);
  }
}

task main()
{
  SetSensor(SENSOR_1, SENSOR_TOUCH);
  start DriveAround;
  until(SENSOR_1);
  stop DriveAround;
  Off(OUT_A+OUT_C);
}
```

What do you think this program will do? Try it!

Tips on using multiple tasks.

- When you press the Run button on the RCX, only `task main` starts automatically, all other tasks have to be started from somewhere in your program.
- To start a task, use `start taskname;` To stop, use `stop taskname;`
- You can name a task anything you want, as long as it starts with a letter and only contains letters, numbers, or the '_' character.
- Stopping a task does not stop the motors. If you want to make sure your robot is stopped after you stop a task, you must use the `Off` command after the `stop taskname;` statement.

What else can you make your robot do?