Matt Buland
Bootstrapping Project #1
Hours worked: 2hrs

Task 5/6:

Question Response: There seems to be one mote at a time that gets a lot of data through, then alternating to the other mote. I'd presume that there's collision happening, and only one mote's data is getting received, stomping out the other's. This is why there's some missing data.

```
void setup() {
   pinMode(13, OUTPUT);
   Serial.begin(57600);
   Serial.setTimeout(1);
}

char receive_byte[1];
int incr;
char recstatus;

void loop() {
  recstatus = Serial.readBytes(receive_byte, 1);
  if (recstatus != 0) {
    if (receive_byte[0] == 'S') {
      incr = 0;
    }
  }

  if (receive_byte[0] == 'S') {
    Serial.print("B ");
    Serial.println(incr);
    incr++;
  }
  delay(1);
}
```

Task 7:

```
void setup() {
   pinMode(13, OUTPUT);
   Serial.begin(57600);
   Serial.setTimeout(1);
}

char receive_byte[3];
int incr = 0;
char recstatus = 0;
char myname = 'B';
int gotime = 0;

void loop() {
  recstatus = Serial.readBytes(receive_byte, 3);
```

```
  if (recstatus == 3) {
    if (receive_byte[0] == 'S' && receive_byte[2] == myname) {
      incr = 0;
      gotime = 1;
    } else if (receive_byte[0] == 'T' && receive_byte[2] == myname) {
      gotime = 0;
    }
  }

  if (gotime == 1) {
    Serial.print(myname);
    Serial.print(" ");
    Serial.println(incr);
    incr++;
  }
  delay(1);
}
```

Task 8:

When timing transmissions, there seems to be relative consistency when transmitting sparse data, one at a time. When doing successive transmissions back-to-back, though, it easily stacks up the transmission times, but up to a maximum. In general, the maximum was about 10x the minimum, and the maximum transmission could be seen by having two transmissions directly back to back. My guess for why this delay exists is that a queue inside the Xbee radio holds a bit of data before it gets sent, and the Serial.write() function doesn't return until the buffer has been cleared (but not necessarily when the transmission was sent), which is why the second time through, the transmission hasn't finished, so it blocks until it finishes, then queues up the next. I could easily be wrong though.

```
void setup() {
   pinMode(13, OUTPUT);
   Serial.begin(57600);
}


char send_byte[100];
char recstatus = 0;
int end_t = 0;
int start_t = 0;
int nbytes = 10;
char rec_byte[1];


void loop() {
  recstatus = Serial.readBytes(rec_byte, 1);


  if (recstatus > 0) {
```

```
    start_t = micros();
    Serial.write((const uint8_t*) send_byte, nbytes);
    end_t = micros();

    if (recstatus > 0) {
      Serial.print("Transmit ");
      Serial.print(nbytes);
      Serial.print(" bytes: ");
      Serial.print(end_t-start_t);
      Serial.println(" microseconds");
    }
  }
}
```