Summary:

This activity will walk students through manual parsing of NMEA sentences. For a review of NMEA sentences, please see Lecture 18 and Appendix J – NMEA Strings.

Materials:

Each student should have the following materials, equipment and supplies:
- Computer with Arduino IDE installed
- USB-AB programming cable
- Arduino Mega microcontroller with assembled Adafruit Ultimate GPS Logger Shield attached

Procedure:

Activity A: Getting Data From Unparsed NMEA Strings – Introduction

1. Open the `GPSParsingIntroduction.io` sketch and save a copy of it as `GPSManualParsing.io`.
2. Edit the setup() function so that only the VTG sentence is being sent by the GPS.
3. Delete the sections in loop() that have information being printed every 5, 7, and 13 seconds. If you used functions to print this information, delete those functions as well. Remove the global variables that were used in these tasks.
4. Edit loop() so that each new VTG sentence is printed to the Serial Monitor. Compare the output with Appendix J. Make sure you understand the information being presented.
5. In order to manually parse the VTG sentence, a few C++ functions will be required. Look up documentation on the `strtok()`, `indexOf()`, and `substring()` functions. Familiarize yourself with these functions.
   a. Hint: A good place to start is googling `Arduino strtok()`.
6. Edit the global variables so that the beginning the sketch looks like Figure 1. Variables will be required for the different data fields in the VTG sentence (Lines 38 – 42).
7. Now, it’s time to actually parse the VTG sentences. The proper way to do this is to create a function that will parse the sentence. Create a function called `parseVTG` that returns nothing and takes a String called `VTG` as an input. Call this function inside `loop()` after the VTG sentence is printed.

8. To parse the sentence inside `parseVTG()`, the `strtok()` function will be used. To use this function, a char array of the VTG sentence needs to be created. Using the `VTG_char` variable previously declared, create the char array from the NMEA sentence String inside `parseVTG()`.
   a. Hint: To create a char array from a String, use the `toCharArray()` function.

9. Next, let’s have the function print out each parsed bit of information. To do this, we need to create a loop. Create the loop shown in Figure 2.
   a. `strtok()` takes a char array and a delimiter and then parses the char array using the delimiter. Each time it’s called, it returns the next data field. This is why a loop is required to get every data field.
10. Upload the sketch. If the sketch doesn’t have any errors, the output on the Serial Monitor should look like Figure 3.

Figure 2: This is an example of what the function to parse a VTG NMEA sentence could look like. Line 159 shows how to convert a string into a char array. The strtok() function requires a char array so this is necessary. The number of data fields in a VTG sentence determines the limits of the for loop.
11. Examining the output on the Serial Monitor, it can be seen that the empty data field for the degrees magnetic was skipped over. This is result of using strtok(), but it is not our desired result. To overcome this, a function needs to be created that will insert a space between two consecutive commas. Create a function called `fillBlankFields` that takes a String as an input and returns another String.

12. The function `indexOf()` returns the first index of the substring that was searched for. If the substring isn’t found in the String, then a -1 is returned. Because of this behaviour, it makes sense to use a while loop. Create the body of `fillBlankFields` using Figure 4.
13. Edit `parseVTG()` so that calling `fillBlankFields()` is performed first. Input the VTG sentence and have the result from `fillBlankFields()` write over the VTG sentence variable.

14. Upload the sketch. The Serial Monitor output should now look like Figure 5.

Figure 4: This is a function that will take a String, find any double commas in the String, and insert a space between the commas. The String returned by this function is identical to the inputted String, unless double commas were found. If the `indexOf()`, `substring()`, and `length()` functions are confusing, look up their documentation. If you are still confused, ask an instructor for guidance.
15. Next, edit the for loop so only the data fields with information are extracted and printed. This means degrees true, degrees magnetic, speed in knots, speed in kilometres per hour, and the mode/checksum. Save this information in the correct variables and print labels to the Serial Monitor. The “$GPVTG,” “T,” “M,” “N,” and “K” do not need to be saved because they do not change. If done properly, the Serial Monitor output should look like Figure 6.

a. Hint: Use if statements within the for loop.
16. Move the information printing from the `parseVTG` function to the main loop. Clean up the `parseVTG` function. If done properly, the Serial Monitor should still look like Figure 6. The cleaned up version of `parseVTG` should look something like Figure 7.

```c
void parseVTG(String VTG) {
    // Function to parse VTG NMEA sentence
    VTG = fillBlankFields(VTG); // Insert a space between any consecutive commas
    VTG.toCharArray(VTG_char, 50); // Transform the VTG String into a char array
    token = strtok(VTG_char, delim); // Get the first data field and print it

    // Loop through the VTG char array and extract the various data fields
    for (int i = 0; i<10; i++) {
        if (i == 1) degreesTrue = token;
        if (i == 3) degreesMagnetic = token;
        if (i == 5) speedKnots = token;
        if (i == 7) speedKmh = token;
        if (i == 9) modeCS = token;
        token = strtok(NULL, ",");
    }
}
```

Figure 7: This is an example of what the final `parseVTG` function could look like. Nothing should be printed inside this function and nothing should be returned. The purpose of this function is to parse the VTG sentence and save the information to the proper global variables. Do not forget to comment your code, comments will make editing and referencing your code later much, much easier.

17. Before proceeding, show your code and Serial Monitor to an instructor.
Activity B: Getting Data From Unparsed NMEA Strings – Part 2

1. If needed, review the last section for how to manually parse a NMEA sentence.

2. The two remaining NMEA sentences are GSA and GSV. Edit `GPSManualParsing.io` to parse the remaining sentences. The end result should be all parsed information being printed to the Serial Monitor, as shown in Figure 8.

   a. Hints: NMEA sentence char arrays need to be big enough to store all data. `strtok()` returns `NULL` when there are no more delimiters. Arrays might be beneficial for storing information; for example, the variables defined for the GSA sentence might look something like Figure 9.

![Figure 8: Shown is the Serial Output of the sketch that parses the GSA, VTG, and GSV NMEA sentences. Depending on the fix quality, there may not be as many GSV sentences. The fewer the number of satellites being tracked, the fewer number of GSV sentences.](image-url)
3. Before proceeding, show your code and Serial Monitor to an instructor.

4. Congratulations! You have successfully completed this activity!