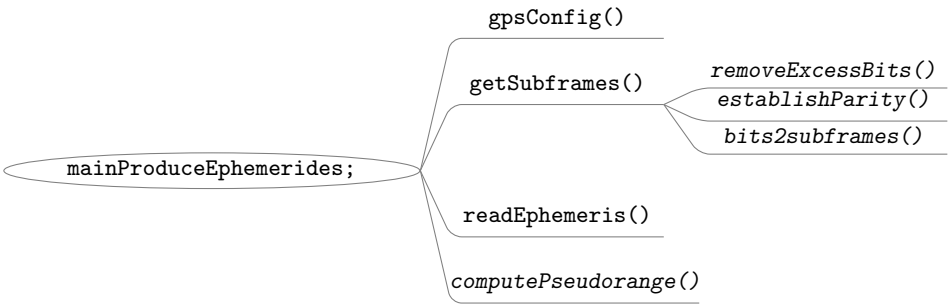
GPS Subframe Detection and Ephemeris Data Extraction

The goal of this assignment is to extract the ephemeris data from the received bits and to compute the pseudorange.

Start by downloading the code framework for this assignment from Moodle[1]. Extract the archive into a directory of your choice (but do not mix them with the files of the previous assignment). The archive file contains the complete decoded bit sequences and corresponding `tau` sequences for all visible satellites.

NOTES

- To get an overall idea of how the program works, read through the provided code before you start writing your own code. Especially, go over the code of `getSubframes`. Note that the sequence of bits is read from files `bitsXX-long.mat` in the `data/` directory, where `XX` is the satellite number. At the same time, have a look at the following graph which reflects the program structure[2]



- You can test your implementation by running `mainProduceEphemerides`.
- All the functions that you write must be saved in the `ephemerides/` directory. In Python, they are located within `utilGpsEphemerides.py`.
- As in the previous assignments, we provide the solutions for every function that you should write. You can use these solutions to test a whole working system, maybe just substituting one function at a time with your implementation. For MATLAB, the script `function_mapper` is used to select for each function whether to use the provided solution or your own implementation.

EXERCISE 1. `getSubframes()` takes as argument a sequence of decoded bits (i.e., the output saved to disk by `decodeSatellites()` from the previous assignment) and returns the subframes and their IDs. `computePseudorange()` determines the pseudorange expressed in meters. Your task is to complete the following four functions seen in class.

(a) `removeExcessBits()`: takes a sequence of ± 1s as returned by `decodeSatellites()`. To complete the function, you must find the position of the subframes in this sequence (by looking for the occurrences of the preamble), and strip off the bits corresponding to incomplete subframes at the beginning and end of the sequence.

`removeExcessBits()` also performs the conversion from the $\{1, -1\}$ representation to a $\{0, 1\}$ representation of the bits. This conversion depends on the sign of the preamble found in the data; recall that when decoding the data, we arbitrarily locked the first bit to 1 or -1, so the decoded bits could all be inverted. Once you find the preambles (or their inverses), you know if you need to invert the mapping or not.

The function also returns the number of bits that were removed from the beginning of the input sequence.

[1] File `gpsEphemerides_assignment.zip`

[2] Names ending with `()` denote functions, and names ending by `;` denote scripts. The functions in italic font are those that you will write as part of the assignment.

Hints.

- The preamble is stored in `gpsc.preamble` as a sequence of 0s and 1s.
- It is not enough to just find one preamble: the preamble is 8 bits long and it is not unlikely that this 8-bit sequence is also present somewhere in the middle of the data sequence. You should use the fact that there is one preamble in every subframe (every 300 bits).
- The result of the correlation operation are real numbers and it can have multiple maxima. If you want to compare two values returned by the correlation, you should first round them.

(b) `establishParity()`: implements the GPS parity check algorithm as seen in class and returns the processed subframe sequence. We provide a skeleton for the function and ask you to complete the missing parts.

(c) `bits2subframes()`: returns a matrix having as its columns the subframes extracted from the parity-checked bits. The function should return the IDs of the subframes as well, so you need to analyze the subframe ID field (stored in left-MSB binary form in bits 50 to 52 of each subframe). Provided that we have enough bits, the output matrix should contain the subframes with ID 1,2,3, in that order (it might contain additional subframes and the first column is not necessarily subframe 1). These are the subframes that we use to obtain the ephemerides.

(d) `computePseudorange()`: computes the pseudorange expressed in meters for a given satellite. It has the following input arguments: the sequence of `tau` values; the index in the vector of received samples that corresponds to the time t^* at which the position (of the receiver) is to be computed; the index of the first bit of the first subframe of that satellite (as returned by `removeExcessBits()`).