Processing FASTA files

Exercise for processing DNA sequences.

- **Contact:** alexander.kanitz@unibas.ch

Exercise 1.1 (3 points)

Write a function `parse_fasta()` that takes a path to a FASTA file as input and returns a tuple of two lists, the first containing sequence headers stripped of the leading `>`, and the second containing the actual sequences.

> **Notes:**
>
> - Please write the parser from scratch and do *not* use existing FASTA parsers, such as the one provided by Biopython.
> - Ensure that wrapped sequences are handled such that fragments of a given sequence are concatenated, without white space, in the order they appear in the file. Make use of the leading `>` character to separate records from each other.
> - Ensure that the number of items in the returned lists correspond to the original number of records in the input file.

Exercise 1.2 (2 points)

Write a function `discard_ambiguous_seqs()` that takes a list of strings as input and returns only those strings that exclusively consist of letters of the "DNA alphabet" (`A`, `C`, `G`, `T`).

> **Notes:**
>
> - Make sure your implementation is case-insensitive, i.e., sequences containing lowercase DNA characters, even if mixed with uppercase characters, are valid as well.

Exercise 1.3 (2 points)

Write a function `nucleotide_frequencies()` that takes a list of strings as input, and which prints out the total frequency of each nucleotide across all input sequences. Use the following example as a template to format your output:

```
A: 0.3
C: 0.21
G: 0.19
T: 0.3
```

> **Notes:**
>
> - Note how numbers are rounded in the example and format decimals printed by your solution in the same manner, i.e., rounded to a single significant digit.

> - The function does not require any specific return value. In case you are not aware of how Python deals with functions without an explicit `return` statement, look up the behavior in relevant documentation.

Exercise 1.4 (3 points)

Write a function `map_reads()` that takes as input two FASTA files, the first containing short read sequences ("query"), and the second containing reference sequences. The function should read the files, discard *query* sequences that contain non-DNA characters, print the nucleotide fractions for both files to the console and returns a dictionary of dictionaries, where the outer dictionary uses the names of query sequences as its keys, and the inner dictionary uses reference sequence names as keys and a list of 1-based indices indicating at which position (counting from left to right) in the reference sequence the query sequence occurs as an exact substring.

Execute the function, passing `sequences.fasta` and `genome.fasta` as input. Inspect the returned "hits" object (the dictionary of dicionaries). Interpret the results in at least 2-3 bullet points. What's special about query sequence `sequence4`?