CS1112 Fall 2014 Project 5 due Thursday 11/6 at 11pm

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not ok for you to see or hear another student's code and it is certainly not ok to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, please seek help from the course staff.

Objectives

Completing this project will solidify your understanding of character array and cell array and give you practice with processing text data files. Another focus is for you to turn a problem statement written in English into a solution written as a MATLAB program. You will *design* the solution to the problem posed. Minimal specifications are given so that you will design the appropriate program structure, including any subfunctions, yourself.

Read carefully and take some time to think about the *design*—don't just jump into coding immediately.

1 Finding Genes

A four-letter alphabet, A, C, T, and G, is used to represent the four nucleotides, Adenine, Cytosine, Thymine, and Guanine, in the DNA of living organisms. There is one (long) sequence of these nucleotides, or bases, for each chromosome, and the set of sequences for all the chromosomes in an organism is called the genome. The genomic sequences for many organisms are now known, including a human genome, which is a sequence of about three *billion* characters!

A gene is a substring of a genome that codes for a specific function in an organism. It is a chain of *codons*, each of which is a triplet of bases that encodes one amino acid (e.g., the codon CAA is the amino acid Glutamine, which plays a role in regulating the acid-base balance in the kidney). In a sequence of bases, the *start codon* ATG marks the beginning of a gene, and one of three *stop codons*, TAG, TAA, or TGA, marks the end of a gene (but the start and stop codons themselves are not part of the gene). One of the first steps in analyzing a genome is to identify its genes.

You will write a program to find all the genes in a partial genomic sequence. Specifically, you will write a function `possibleGenes` such that the statement

```
n= possibleGenes('data.txt', 'result.txt')
```

reads a DNA data file called `data.txt` in the current directory, identifies all the genes in the sequence in the data file, writes the genes to a file called `result.txt`, and stores in variable `n` the number of genes found. *This is the only specification* in addition to the file formats that will be given below following the example. You will design the entire program and implement it following good programming style. Specifically, your gene finding algorithm must be clear—encapsulate specific, detailed tasks as subfunctions so that the structure of your gene finding algorithm is easy to understand.

1.1 Example

Consider the following string, which is not a real genomic sequence (at least not yet!). The sequence

AATGCCCATGGGGTATGTGTGAATGACATGCACTAACATGAAGTTGTAGGTTTAG

encodes four genes, which are underlined below:

AATGCCCATGGGGTATGTGTGAATGACATG<u>CAC</u>TAACATGAAGTTGTAGGTTTAG
AATGCCCATG<u>GGGTATGTG</u>TGAATGACATGCACTAACATGAAGTTGTAGGTTTAG
AATGCCCATGGGGTATGTGTGAATGACATGCACTAACATG<u>AAGTTG</u>TAGGTTTAG
AATGCCCATGGGGTATG<u>TGTGAA</u>TGACATGCACTAACATGAAGTTGTAGGTTTAG

Note the following features:

- A gene has a non-zero length that is a multiple of 3 (since a codon is a triplet of bases).
- A gene is preceded by the start codon and is followed by a stop codon.
- A gene, i.e., a sequence of codons, does not include a start codon or a stop codon.
- It is possible for the identified genes to "overlap" since there are three "reading frames" within which one can identify codons (again because a codon is a triplet). The first "frame" starts at the beginning of the sequence. Below, the first 36 bases of the example sequence are shown, grouped in triplets with zero offset:
  AAT—GCC—CAT—GGG—GTA—TGT—GTG—AAT—GAC—ATG—CAC—TAA
  With this frame, we find the gene CAC. Next we show the same 36 bases with a reading frame offset by 1:
  A—ATG—CCC—ATG—GGG—TAT—GTG—TGA—ATG—ACA—TGC—ACT—AA
  With this second frame, we find the gene GGGTATGTG. Finally we show the 36 bases with a reading frame offset by 2:
  AA—TGC—CCA—TGG—GGT—ATG—TGT—GAA—TGA—CAT—GCA—CTA—A
  With this third frame, we find the gene TGTGAA.

1.2 File Format

The data files `humanC5_?.txt` where `?` is 1, 2, or 3, are portions of the human chromosome 5 nucleotide sequence downloaded from the data bank of the European Bioinformatics Institute. The sequences in the files are approximately 100, 1000, and 10000 bases long, respectively. The files contain ASCII characters (plain text). Each file begins with a line of text that identifies the sequence but *is not* part of the nucleotide sequence. The remaining lines in the file contain the sequence and are of varying lengths. Your code should work with the given files—do not modify the files in any way!

Your program writes the resulting gene data file in the following way: (1) write an ASCII (plain text) file; (2) each line corresponds to one gene and is exactly the length of the gene—no extra spaces; (3) there are as many lines as there are genes; and (4) the file is named as specified by the function call to `possibleGenes`.

1.3 Hints

- Break down the problem! There are three main tasks: find genes given a string, read a file, and write a file. Don't try to do everything at once.
- Creating a vector of characters is a good intermediate step between reading data and finding genes. Similarly, creating a cell array of genes is a reasonable intermediate step before writing the final data file.
- Subfunctions are your friends! Although it takes time to specify subfunctions, once you have subfunctions for specific tasks your overall algorithm, and main program development, is simplified. Tasks that are great candidates for subfunctions include checking whether a triplet is a start codon, checking whether a triplet is a stop codon, and identifying genes in a sequence with zero offset.
  During program development, write independent functions instead of subfunctions! The reason is that an independent function can be tested individually just by calling it in the Command Window, without involving other functions or tasks. *After* you have solved the entire problem, copy over all the functions into one file `possibleGenes.m` (`possibleGenes` is the main function).
- Use the example sequence above in developing your gene finding algorithm. Copy it and paste it into a MATLAB script file that you save so that you don't need to type it over and over again. This way you can develop your gene finding algorithm (hopefully as a (sub)function) without worrying about text file processing.
- One of the three given data files doesn't encode a gene—be sure that your program ends appropriately and not in error. If a gene isn't found, `possibleGenes` returns the value zero and it is up to you whether to write an empty file or not write a file at all.

Submit your file `possibleGenes.m` on CMS.