Homework 2: Sequence Alignment

CSCI 5481, Computational Techniques for Genomics
University of Minnesota
Instructor: Dan Knights

Instructions

- Please turn this assignment in on the course web page.
- There are multiple files to turn in. All text and code should be placed into a single folder with a name like *lastname_exerciseXX*. The folder should then be compressed and submitted as a single archive (.zip or .tgz)
- You must do this work on your own, although you are encouraged to have general discussions with other students. The work you turn in must be your own. Your code will be checked for overlap and for surprising idiosyncrasies in common with other submissions.
- Please write the names of anyone with whom you discussed this assignment at the top of your assignment.
- Please include copious comments in your code. Full credit will only be given for code that is fully commented, meaning that every line that is not completely obvious needs a comment. Partial credit may be given for broken/non-functioning code if the code is well-commented.
- You may use any programming language you wish.
- **You may use ChatGPT or other AI models as a collaborator.** You may copy code directly from the AI, if you cite the source properly. You must use comments to mark which sections of your code were generated by the AI, and include comments where you made modifications, as in this example:

  ```
  ##### Begin code from Chat-GPT 4.0 #####
  print("some code from AI.")
  a = [1,2,3,4]
  # modified to remove brackets around the 5
  a = a.append(5)
  ##### End code from Chat-GPT 4.0 #####
  ```

  Note that you could, in theory, generate the entire answers using AI. However for complex tasks, most of the time, the AI will make critical mistakes. You might spend more time trying to figure out the AI's mistakes than it would take you to write the code yourself. Also, you will probably learn more if you write and design the code yourself. Therefore, please use AI bots with caution.

Background

This homework assignment is implementation of the standard Needleman-Wunsch algorithm and application of the algorithm to align SARS-CoV-2 spike proteins from mRNA vaccines and the original SARS genome, first in nucleotide space, and then in amino acid space.

Dataset

Download and extract the folder containing the SARS-Cov-2 spike protein RNA sequence from the reference genome and the mRNA sequences from the initial Pfizer and Moderna COVID-19 vaccines (on Canvas here).

Input and Output Format:

The command line for calling your program should be of the form: `programname.py seq1.fasta seq2.fasta output.txt`. Output file should contain both the alignment score for this pair of sequences and the actual alignment itself printed with gaps shown using “_”. You ~~may~~ must use command-line flags to label your parameters, like this:

```
python3 aligner.py -q pfizer_mrna.fna -r sars_spike_protein.fna -o
question_2_output.txt -g -2 -p -1 -m 1
```

Notes: -g = gap penalty; -p = mismatch penalty; -m = match score

The output alignment file should contain 6 lines in this format:

1. Alignment score
2. Seq1 header or label
3. Seq1 with gaps shown as “_”
4. An alignment visualization, with “|” for match, “x” for mismatch, and “ ” for gap
5. Seq2 with gaps shown as “_”
6. Seq2 header or label

For example:

```
23 (note, this is made up)
>SARS-Cov-2_reference_genome_spike_protein
__C_GATT_TCGGG...
  |  ||| |xx||
AGCA_ATTCTTCGG...
>Pfizer_mRNA_vaccine
```

Tasks

1. (30 points): Implement the Needleman-Wunsch algorithm for global alignment (don't forget your comments), including an option for un-penalized start and end gaps in both sequences. You can make the penalties hard-coded or commandline options. Commandline will probably make it easier for you to debug/do the rest of the assignment (see argparse in python, e.g. for adding commandline flags in a graceful way – this is optional).

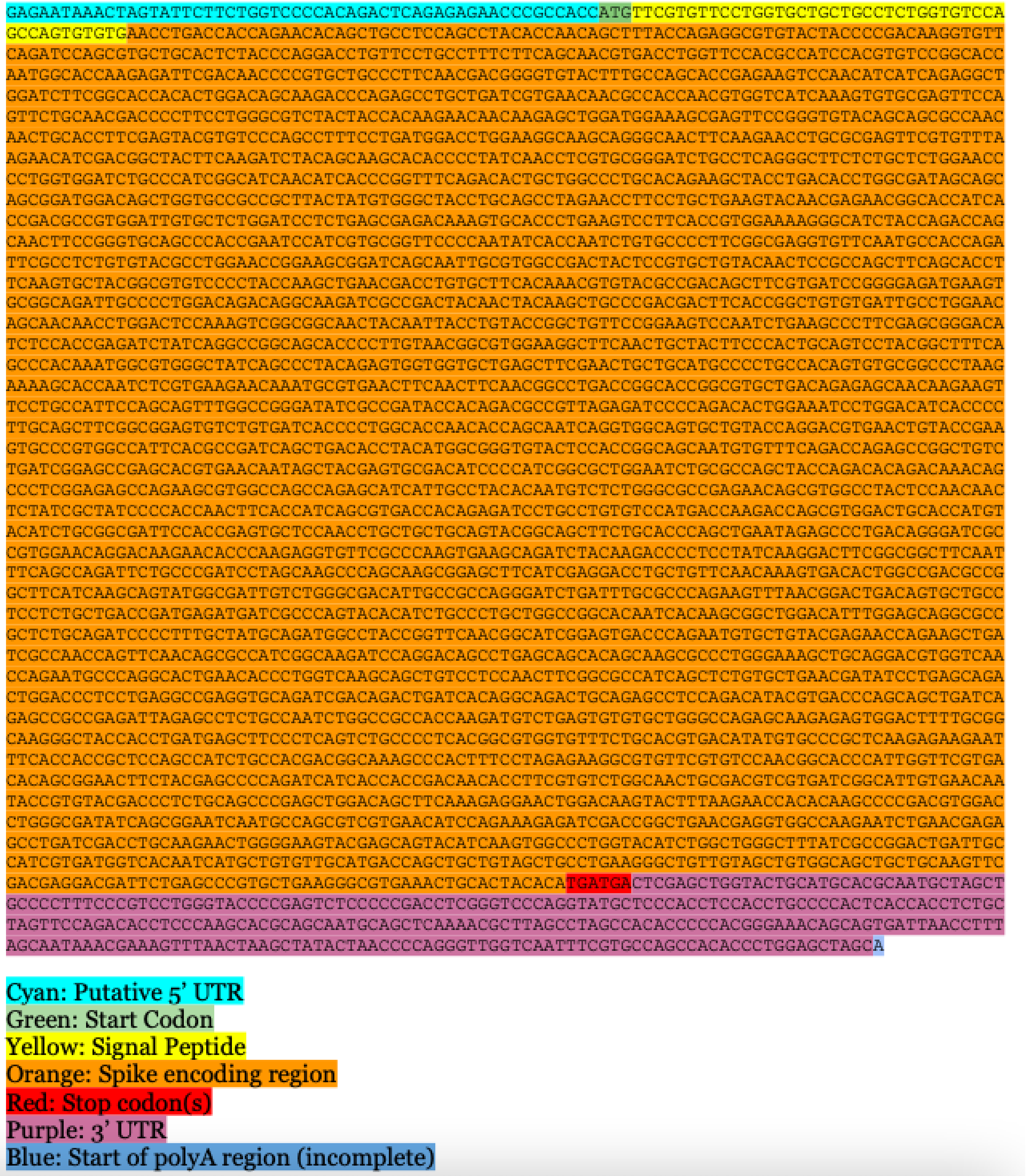
2. (10 points): Align the Pfizer mRNA vaccine to the SARS-Cov-2 reference spike protein sequence using global alignment and a fixed gap penalty, as follows:
    a. Fixed gap penalty: -2 per gapped position
    b. Mismatch penalty: -1
    c. Match score: 1

3. (10 points): Align the Pfizer mRNA vaccine to the SARS-Cov-2 reference spike protein sequence using global alignment and a fixed gap penalty, but with no penalty for start/end gaps on either sequence, as follows:
    a. Initial or terminal gap penalty: 0
    b. Other gap penalty: -2 per gapped position
    c. Mismatch penalty: -1
    d. Match score: 1

4. (10 points): Compare your alignments from #2 and #3. In what way did the free start and end gaps improve the alignment?

5. (10 points): Based on this annotated image of the full sequence of Pfizer mRNA vaccine, why did it make sense to ignore start gaps and end gaps in your alignment in #3?

Figure 1: Spike-encoding contig assembled from BioNTech/Pfizer BNT-162b2 vaccine.



```
GAGAATAAACTAGTATTCTTCTGGTCCCCACAGACTCAGAGAGAACCCGCCACCATGTTCGTGTTCCTGGTGCTGCTGCCTCTGGTGTCCA
GCCAGTGTGTGAACCTGACCACCAGAACACAGCTGCCTCCAGCCTACACCAACAGCTTTACCAGAGGCGTGTACTACCCCGACAAGGTGTT
CAGATCCAGCGTGCTGCACTCTACCCAGGACCTGTTCCTGCCTTTCTTCAGCAACGTGACCTGGTTCCACGCCATCCACGTGTCCGGCACC
AATGGCACCAAGAGATTCGACAACCCCGTGCTGCCCTTCAACGACGGGGTGTACTTTGCCAGCACCGAGAAGTCCAACATCATCAGAGGCT
GGATCTTCGGCACCACACTGGACAGCAAGACCCAGAGCCTGCTGATCGTGAACAACGCCACCAACGTGGTCATCAAAGTGTGCGAGTTCCA
GTTCTGCAACGACCCCTTCCTGGGCGTCTACTACCACAAGAACAACAAGAGCTGGATGGAAAGCGAGTTCCGGGTGTACAGCAGCGCCAAC
AACTGCACCTTCGAGTACGTGTCCCAGCCTTTCCTGATGGACCTGGAAGGCAAGCAGGGCAACTTCAAGAACCTGCGCGAGTTCGTGTTTA
AGAACATCGACGGCTACTTCAAGATCTACAGCAAGCACACCCCTATCAACCTCGTGCGGGATCTGCCTCAGGGCTTCTCTGCTCTGGAACC
CCTGGTGGATCTGCCCATCGGCATCAACATCACCCGGTTTCAGACACTGCTGGCCCTGCACAGAAGCTACCTGACACCTGGCGATAGCAGC
AGCGGATGGACAGCTGGTGCCGCCGCTTACTATGTGGGCTACCTGCAGCCTAGAACCTTCCTGCTGAAGTACAACGAGAACGGCACCATCA
CCGACGCCGTGGATTGTGCTCTGGATCCTCTGAGCGAGACAAAGTGCACCCTGAAGTCCTTCACCGTGGAAAAGGGCATCTACCAGACCAG
CAACTTCCGGGTGCAGCCCACCGAATCCATCGTGCGGTTCCCCAATATCACCAATCTGTGCCCCTTCGGCGAGGTGTTCAATGCCACCAGA
TTCGCCTCTGTGTACGCCTGGAACCGGAAGCGGATCAGCAATTGCGTGGCCGACTACTCCGTGCTGTACAACTCCGCCAGCTTCAGCACCT
TCAAGTGCTACGGCGTGTCCCCTACCAAGCTGAACGACCTGTGCTTCACAAACGTGTACGCCGACAGCTTCGTGATCCGGGGAGATGAAGT
GCGGCAGATTGCCCCTGGACAGACAGGCAAGATCGCCGACTACAACTACAAGCTGCCCGACGACTTCACCGGCTGTGTGATTGCCTGGAAC
AGCAACAACCTGGACTCCAAAGTCGGCGGCAACTACAATTACCTGTACCGGCTGTTCCGGAAGTCCAATCTGAAGCCCTTCGAGCGGGACA
TCTCCACCGAGATCTATCAGGCCGGCAGCACCCCTTGTAACGGCGTGGAAGGCTTCAACTGCTACTTCCCACTGCAGTCCTACGGCTTTCA
GCCCACAAATGGCGTGGGCTATCAGCCCTACAGAGTGGTGGTGCTGAGCTTCGAACTGCTGCATGCCCCTGCCACAGTGTGCGGCCCTAAG
AAAAGCACCAATCTCGTGAAGAACAAATGCGTGAACTTCAACTTCAACGGCCTGACCGGCACCGGCGTGCTGACAGAGAGCAACAAGAAGT
TCCTGCCATTCCAGCAGTTTGGCCGGGATATCGCCGATACCACAGACGCCGTTAGAGATCCCCAGACACTGGAAATCCTGGACATCACCCC
TTGCAGCTTCGGCGGAGTGTCTGTGATCACCCCTGGCACCAACACCAGCAATCAGGTGGCAGTGCTGTACCAGGACGTGAACTGTACCGAA
GTGCCCGTGGCCATTCACGCCGATCAGCTGACACCTACATGGCGGGTGTACTCCACCGGCAGCAATGTGTTTCAGACCAGAGCCGGCTGTC
TGATCGGAGCCGAGCACGTGAACAATAGCTACGAGTGCGACATCCCCATCGGCGCTGGAATCTGCGCCAGCTACCAGACACAGACAAACAG
CCCTCGGAGAGCCAGAAGCGTGGCCAGCCAGAGCATCATTGCCTACACAATGTCTCTGGGCGCCGAGAACAGCGTGGCCTACTCCAACAAC
TCTATCGCTATCCCCACCAACTTCACCATCAGCGTGACCACAGAGATCCTGCCTGTGTCCATGACCAAGACCAGCGTGGACTGCACCATGT
ACATCTGCGGCGATTCCACCGAGTGCTCCAACCTGCTGCTGCAGTACGGCAGCTTCTGCACCCAGCTGAATAGAGCCCTGACAGGGATCGC
CGTGGAACAGGACAAGAACACCCAAGAGGTGTTCGCCCAAGTGAAGCAGATCTACAAGACCCCTCCTATCAAGGACTTCGGCGGCTTCAAT
TTCAGCCAGATTCTGCCCGATCCTAGCAAGCCCAGCAAGCGGAGCTTCATCGAGGACCTGCTGTTCAACAAAGTGACACTGGCCGACGCCG
GCTTCATCAAGCAGTATGGCGATTGTCTGGGCGACATTGCCGCCAGGGATCTGATTTGCGCCCAGAAGTTTAACGGACTGACAGTGCTGCC
TCCTCTGCTGACCGATGAGATGATCGCCCAGTACACATCTGCCCTGCTGGCCGGCACAATCACAAGCGGCTGGACATTTGGAGCAGGCGCC
GCTCTGCAGATCCCCTTTGCTATGCAGATGGCCTACCGGTTCAACGGCATCGGAGTGACCCAGAATGTGCTGTACGAGAACCAGAAGCTGA
TCGCCAACCAGTTCAACAGCGCCATCGGCAAGATCCAGGACAGCCTGAGCAGCACAGCAAGCGCCCTGGGAAAGCTGCAGGACGTGGTCAA
CCAGAATGCCCAGGCACTGAACACCCTGGTCAAGCAGCTGTCCTCCAACTTCGGCGCCATCAGCTCTGTGCTGAACGATATCCTGAGCAGA
CTGGACCCTCCTGAGGCCGAGGTGCAGATCGACAGACTGATCACAGGCAGACTGCAGAGCCTCCAGACATACGTGACCCAGCAGCTGATCA
GAGCCGCCGAGATTAGAGCCTCTGCCAATCTGGCCGCCACCAAGATGTCTGAGTGTGTGCTGGGCCAGAGCAAGAGAGTGGACTTTTGCGG
CAAGGGCTACCACCTGATGAGCTTCCCTCAGTCTGCCCCTCACGGCGTGGTGTTTCTGCACGTGACATATGTGCCCGCTCAAGAGAAGAAT
TTCACCACCGCTCCAGCCATCTGCCACGACGGCAAAGCCCACTTTCCTAGAGAAGGCGTGTTCGTGTCCAACGGCACCCATTGGTTCGTGA
CACAGCGGAACTTCTACGAGCCCCAGATCATCACCACCGACAACACCTTCGTGTCTGGCAACTGCGACGTCGTGATCGGCATTGTGAACAA
TACCGTGTACGACCCTCTGCAGCCCGAGCTGGACAGCTTCAAAGAGGAACTGGACAAGTACTTTAAGAACCACACAAGCCCCGACGTGGAC
CTGGGCGATATCAGCGGAATCAATGCCAGCGTCGTGAACATCCAGAAAGAGATCGACCGGCTGAACGAGGTGGCCAAGAATCTGAACGAGA
GCCTGATCGACCTGCAAGAACTGGGGAAGTACGAGCAGTACATCAAGTGGCCCTGGTACATCTGGCTGGGCTTTATCGCCGGACTGATTGC
CATCGTGATGGTCACAATCATGCTGTGTTGCATGACCAGCTGCTGTAGCTGCCTGAAGGGCTGTTGTAGCTGTGGCAGCTGCTGCAAGTTC
GACGAGGACGATTCTGAGCCCGTGCTGAAGGGCGTGAAACTGCACTACACATGATGACTCGAGCTGGTACTGCATGCACGCAATGCTAGCT
GCCCCTTTCCCGTCCTGGGTACCCCGAGTCTCCCCCGACCTCGGGTCCCAGGTATGCTCCCACCTCCACCTGCCCCACTCACCACCTCTGC
TAGTTCCAGACACCTCCCAAGCACGCAGCAATGCAGCTCAAAACGCTTAGCCTAGCCACACCCCCACGGGAAACAGCAGTGATTAACCTTT
AGCAATAAACGAAAGTTTAACTAAGCTATACTAACCCCAGGGTTGGTCAATTTCGTGCCAGCCACACCCTGGAGCTAGCA
```

Cyan: Putative 5’ UTR
Green: Start Codon
Yellow: Signal Peptide
Orange: Spike encoding region
Red: Stop codon(s)
Purple: 3’ UTR
Blue: Start of polyA region (incomplete)

https://raw.githubusercontent.com/NAalytics/Assemblies-of-putative-SARS-CoV2-spike-encoding-mRNA-sequences-for-vaccines-BNT-162b2-and-mRNA-1273/main/Assemblies%20of%20putative%20SARS-CoV2-spike-encoding%20mRNA%20sequences%20for%20vaccines%20BNT-162b2%20and%20mRNA-1273.docx.pdf

6. (5 points): How many mismatches (counting gaps as a mismatch if any) are there between the real spike protein and the Pfizer version, in the coding portion of the RNA sequences?

7. (5 points) : Use the codon table as in Homework 1 (e.g. use this one, or use an external package) to translate the SARS-CoV-2 spike protein and the Pfizer mRNA to amino acid sequences. Call these files `sars_spike_protein.aa` and `pfizer_mrna.aa`. For the Pfizer vaccine, start with the first start codon (ATG or methionine) and end with the first stop codon.

8. (5 points) Use your code to align the SARS and Pfizer amino acid sequences (same gap/match/mismatch penalty as #2 – note that start/end gaps don't end up mattering because the sequences are now the same length).

9. (5 points) How many mismatches are there in the two amino acid sequences (counting gaps as a mismatch if any)? What exactly is different between the amino acid sequence of the vaccine and the amino acid sequence of the real spike protein? Hint: there should not be any differences at the start or end of the alignment because both the real SARS sequence and the coding portion of the Pfizer sequence start with a start codon and end with a stop codon. There is a very small number of differences, somewhere in the middle of the protein.

10. (5 points) Describe why your findings in #9 make sense in the context of this article: https://cen.acs.org/pharmaceuticals/vaccines/tiny-tweak-behind-COVID-19/98/i38

11. (5 points) Why did the vaccine makers introduce so many synonymous mutations into the vaccine? Why didn't they just copy the spike protein sequence exactly (apart from the amino acids discussed in question 10)? Hint: Calculate the GC content of the real spike protein and the coding portion of the Pfizer vaccine mRNA. A quick Google search should find you some possible answers. Another hint: recall from the intro to k-mers in class that we discussed "codon bias" in different species. A Google search regarding "codon bias" in the Pfizer COVID vaccine should also provide some useful papers. If you're having trouble finding answers, this article may help.

12. (2 bonus points): Implement an affine (linear) gap penalty with -10 penalty for starting a gap and -2 penalty for continuing a gap, mismatch penalty -1, match score +1. Allow unlimited start-gaps and end-gaps as in #3 above. For example, a gap of length 4 would be -10 -2 -2 -2 = -16 total penalty. Align the Moderna and Pfizer vaccines with and without the affine gap penalty, and describe the difference between the alignments. Are there any gaps inside the coding region without the affine gap penalty? Are there any gaps inside the coding region with the affine gap penalty? Why does this make sense?

~~Deliverables~~

1. ~~Source file (your code). Please note in your code the names of the people with whom you discussed the assignment.~~
2. ~~Readme file (text). The readme file should contain instructions on how to compile and run the program.~~
3. ~~Alignment results for questions 2, 3, 8 (and bonus, optionally)~~
4. ~~A word file or PDF with your responses to the other questions.~~

Deliverables

1. Please submit to the “HW2 Programming” assignment on Gradescope with the following items
    a. A python file titled **`aligner.py`**, and any other python files from which **`aligner.py`** imports packages, etc.
        i. Please note in your code the names of the people with whom you discussed the assignment.
        ii. If you are not using python, please submit your non-python code instead, and a readme file (text). The readme file should contain instructions on how to compile and run the program. We will download and manually run + grade your code.
    b. The amino acid sequence files that you generated: **`pfizer_mrna.aa`** and **`sars_spike_protein.aa`**
    c. You don’t need to submit your alignment files, the autograder will run your code and generate the alignment outputs, and will verify that they are correct.
    d. The autograder will run the following commands for each question (your code must support them) and verify the output is correct. To add flag parsing, consider using argparse described here or see Appendix below.
        1. Q2: `python3 aligner.py -q pfizer_mrna.fna -r sars_spike_protein.fna -o question_2_output.txt -g -2 -p -1 -m 1`
            a. Notes: -g = gap penalty; -p = mismatch penalty; -m = match score
        2. Q3: `python3 aligner.py -q pfizer_mrna.fna -r sars_spike_protein.fna -o question_3_output.txt -g -2 -p -1 -m 1 --ignore_outer_gaps`
        3. Q8: `python3 aligner.py -q pfizer_mrna.aa -r sars_spike_protein.aa -o question_8_output.txt -g -2 -p -1 -m 1`
        4. Q12 (bonus): `python3 aligner.py -q pfizer_mrna.fna -r moderna_mrna.fna -o question_12_output_affine.txt -g -2 -p -1 -m 1 -s -10 --ignore_outer_gaps` and `python3 aligner.py -q pfizer_mrna.fna -r moderna_mrna.fna -o question_12_output_no_affine.txt -g -2 -p -1 -m 1 --ignore_outer_gaps`
            a. Note: -s = starting/opening a gap penalty (for affine penalty)
2. Please submit to the “**HW2 Written.pdf**” assignment on Gradescope the following items
    a. A PDF with your responses to the other questions (4, 5, 6, 9, 10, 11, 12)

To summarize, your submission should include the following four files: **aligner.py, pfizer_mrna.aa, sars_spike_protein.aa, HW2 Written.pdf,** README.txt (optional)

Appendix (using argparse)

See the following code sample below for using argparse:

```
import argparse
```

```
# set up argument parser
parser = argparse.ArgumentParser()
parser.add_argument('-q', '--query', required=True, type=str)
parser.add_argument('-r', '--reference', required=True, type=str)
parser.add_argument('-o', '--output', required=True, type=str)
parser.add_argument('-g', '--gap_penalty', required=True, type=int)
parser.add_argument('-p', '--mismatch_penalty', required=True, type=int)
parser.add_argument('-m', '--match_score', required=True, type=int)
parser.add_argument('--ignore_outer_gaps', action='store_true')
parser.add_argument('-s', '--affine_gap_penalty', required=False, default=0,
type=int)

# parsing arguments to get variables (can rename variables anything you choose,
this just illustrates how to get contents of the parser)
args = parser.parse_args()
q_file, ref_file, out_file = args.query, args.reference, args.output
d, mismatch, match = args.gap_penalty, args.mismatch_penalty, args.match_score
startend_tf = args.ignore_outer_gaps
a = args.affine_gap_penalty
```