Aim: In this assignment, given **n** DNA sequences, $2 \leq n \leq 10$, in a **single aln-formatted** (multiple alignment formatted) file, we ask to implement **sequence to profile alignment**. You may utilize the **naïve** implementation of Needleman-Wunsch from the previous assignment or you can write from the scratch. You will take gap penalty, match score, and mismatch penalty via parameters. The length of the aligned sequences might be at most 500 characters. For the sake of simplicity, we give the alignment file (.aln) described below.

Command line examples: Be sure that your code works using the following command:

```
alignSeqToProfile --fasta seq.fasta --aln aligned_sequences.aln --out seq.aln
--gap ${gap_penalty} --match ${match_score} --mismatch ${mismatch_penalty}
```

Input:

- −−**aln** Only one aln-formatted file containing all given alignments "aligned_sequences.aln", which contains n DNA sequences line-by-line. In the example below, alignment is created randomly for n = 5:

  ```
  sequence1 ATAC---CTAATTGGAGATGATCAAATTTATAAT
  sequence2 TTAT---CTAATTGGCGACGATCAAATTTATAAT
  sequence3 ATAT---CTAATTGGTGATGATCAAATTTATAAT
  sequence4 ATCA---TTAATTGGAGATGATCAATCCTAATGA
  sequence5 CTGTACTTTTATTGGTGATAGTCAAATCTATAAT
  ```

- −−**fasta** Sequence fasta file to be aligned to the given profile

  ```
  > sequence
  CTAGATAATTGGAGATGATCAAATTTATAT
  ```

Parameters:

- −−**gap** gap penalty score
- −−**match** matching score
- −−**mismatch** mismatch penalty score

Output: (Only for the given specific example below: (match, mismatch, gap) = (1, -1, -2))

- −−**out:** sequence.aln

  ```
  sequence1   ATAC---CTAATTGGAGATGATCAAATTTATAAT
  sequence2   TTAT---CTAATTGGCGACGATCAAATTTATAAT
  sequence3   ATAT---CTAATTGGTGATGATCAAATTTATAAT
  sequence4   ATCA---TTAATTGGAGATGATCAATCCTAATGA
  sequence5   CTGTACTTTTATTGGTGATAGTCAAATCTATAAT
  sequence    CTAG---ATAATTGGAGATGATCAAATTTATAAT
  ```

Notes:

- You must write your code yourself. Sufficient evidence of plagiarism will be treated the same as for plagiarism or cheating.
- Non-compiling submissions will not be evaluated.
- Your code will be compiled into a **single binary** using the Makefile. If scripting languages are used, a single wrapper script should be provided.
- Do not submit the program binary. You must submit the following items:
  - All **"source"** files.
  - A script to compile the source code and produce the binary (Makefile), **if required**.
  - A README.txt file that describes how the compilation progress works, **if required**.
- Create a directory, of which format is "surname_name_hw4", put all required files into that directory, and then zip it. You will have a give a single zipped file, 'surname_name_hw4.zip'.
- **Submit your code through the Moodle page. DO NOT EMAIL.**
- C / C++, Python, Java will be used as programming language. STL is allowed. **You may reuse code(s) from your previous homework assignments.** Make sure your code compiles and works in Linux systems (gcc compiler for C/C++).
- All submissions must be made by 23:59, November 30, 2019. Late submissions will lose 20 points for each additional day. Three or more days of delay will result in a zero grade.
- The overall fastest implementation wins. **Bonus** will be given for the fastest.