Bioinformatics Coursework 1

Genome Assembly

Note: Third year students should attempt tasks 1 and 2, whilst fourth year students should attempt tasks 1, 2 and 3.
Third year students will be marked out of 100 – each task being worth 50%
Fourth year students will be marked out of 150 – each task being worth $33^{1}/_{3}$%

The most common approach to DNA sequencing is the shotgun approach. In this case a sequence is broken up into a number of sorter sequences, each typically only a few hundred base pairs long. These shorter sequences can then be assembled into the original DNA sequence on a computer. Note that due to the process of preparing these shorter sequences they often contain overlapping sections and many duplicates.
One of the approaches for performing genome assembly uses a known prior (template) sequence which closely matches the sequence you are trying to assemble. Often described as mapping. For example, you could assemble a chimpanzee sequence using the equivalent human sequence as they are 99% identical.
The process works by taking each short sequence (generated using the shotgun approach) and performing a local sequence alignment with the template sequence. This will give a position of where the shorter sequence will be positioned within the whole (assembled) sequence.

Task 1: Producing a local sequence alignment code (mark: 50)

You are given the code for a global alignment – the Needleman-Wunch algorithm. You should modify this code to become a Smith-Waterman local alignment algorithm. This should be provided as a method within your code that can be called multiple times.
The original code will read a file containing two lines and perform a global alignment on the first two lines. Your modified code should read in two lines at a time from the file and perform a local alignment on each of these pairs of lines.

Marks will be allocated as follows:
Correctness of code: 20
Efficiency of code: 10
Clean and clear code implementation: 10
Correctly computing the results for a set of unseen inputs: 10
You should submit this code separately as part of your submission. The code must be provided in a form that can be executed (i.e. plain ASCII source code).

Task 2: Producing a genome assembly code (mark: 50)

Extend your code from task 1 to read in an initial template sequence followed by any number of shorter sequences which will be locally aligned with the template sequence allowing an entire new sequence to be assembled (i.e. your code will assemble all but the first line of the file into a new sequence). The output should be just the newly constructed sequence.

Marks will be allocated as follows:
Correctness of code: 20
Efficiency of code: 10
Clean and clear code implementation: 10
Correctly computing the results for a set of unseen inputs: 10
You should submit this code separately as part of your submission. The code must be provided in a form that can be executed (i.e. plain ASCII source code).

Please turn over

Task 3: Only for 4th year students: Other genome assembly approaches (mark: 50)

A number of alternate 'de novo' approaches to genome assembly exist – which do not require a previously sequenced genome to use as a template. These include: Celera, Arachne and SGA. A non-exhaustive list of the different algorithms can be found at: https://en.wikipedia.org/wiki/Sequence_assembly.

You should select **one** approach (either one from the Wikipedia page or one from another source) and write a short (no more than 3 pages) discussion of your chosen approach. This will be marked as follows:

A clear description of how the approach works: 20
The strengths of the particular approach: 10
The weaknesses of the particular approach: 10
References to appropriate academic literature: 10