Note: This is part two of a two part assignment. You will use the code you developed for part one of this assignment, however, the main deliverable for this assignment will be a short paper (although you will still turn in your code as well). This assignment is due by **11:55pm on Mon, April 13**. This part of the assignment is worth **20** points. If you choose, you may work with a partner on this assignment. You must submit your short paper and any other requested files as a zip file to Moodle.

1 Project Overview

In this project we will explore the process of genome assembly. Genome assembly of real DNA sequencing data is still a challenging area where many researchers are currently working. In particular, this will be a **reflective assignment** where your task is to use your De Bruijn graph code to build an assembler. Your job is to reflect on what you try and why it does or doesn't work. Your job is NOT to build a great assembler (a very challenging task. In fact, **no part of your grade is dependent on how well your assembler does.**

As an overview, in part two of this assignment you will do the following:

- You will modify your existing De Bruijn code to also produce an assembly and report the N50 and L50 scores for that assembly.
- You will write a short 3 page paper describing what you tried and why it did or didn't work well when creating an assembly. The majority of points for this assignment will be dedicated to this paper.

De Bruijn Starting Code

I expect that students will start work from the code they submitted for HW5. If you had issues with your HW5 code you have two options. (1) Talk to me early and I can help you to resolve the issues. (2) Request (via Slack) that I send you my code (written in Python) for HW5. You can use this code as your starting point.

Any data files mentioned below can be downloaded from Moodle: **HW6 Files**. This directory also contains examples of any file formats mentioned below. Lastly, **please read this whole document carefully before beginning.**

2 Project Specifications

Important: I do NOT expect you to be able to assemble "full size" genomes as that would require more RAM than is available on most personal machines. You should be able to handle data created for the `sample.fasta` file provided.

2.1 Assembler Specifications

You will need to provide the following Python script: `assemble.py` that constructs a De Bruijn Graph from a set DNA sequencing reads and then uses that graph to create an assembly which will be saved in a filed called `contigs.txt`.

`assemble.py` will take in the following parameters (in this order):

- Reads file (string) - A file of reads, as output by `simulate.py`.

- k (int) - the size of k-mer to use when building a De Bruijn graph.

`assemble.py` will construct a de Bruijn graph for the specified k-mer size from the set of reads contained in the supplied `reads.txt` file and then use this graph to produce an assembly (set of contigs). More specifically, your code should complete the following tasks.

- The program will create a file in the current directory called `contigs.txt`. Each line of the file will contain the sequence of an assembled contig. This file should look like the following (see the data directory for an example file) which shows 3 contigs.

  ```
  TAGCACCACTTCTGCGACCCAAGTTG
  TCCGATCCTATATTACGACTTCGGGAAGGGGTTCGCAAGTCCCACCCTAAACGATGTTGAAGGCTCAGG
  TTACACAGGCACAAGTACTATATATACGTGT
  ```

- Your code should print to the screen **only** the N50 and L50 scores for the produced assembly. Please no longer print the size of the De Bruijn graph in the code you submit. For example, the output of your program may look like the following.

  ```
  $ python assemble.py reads.txt 3

  N50 for assembly: 6
  L50 for assembly: 2
  ```

2.2 How to Design an Assembler?

How you choose to design your assembler is completely up to you as long as it follows the exact specifications outlined in the previous section. For example, the most naive assembler would just return the exact set of reads that were given as input. According to our definition of assembly, this is a valid assembly. Here are a few suggestions of ways you might get inspiration:

1. **Look back the class notes.** We briefly talked about a lot of approaches you could try to dig into more deeply. You could try to find the closest thing to an Eulerian path in the grap. You could do some graph modifications first that might help. You could just try to string reads together in the graph. These are all ideas, but none of them are required.
2. **Read a paper for inspiration.**. You are more that welcome to look at what other people have done and to use this as inspiration. If you do this approach, please make sure you cite the paper in your final paper.
3. **Use your intuition.**. You are more than welcome to try out your own approach. Again, whatever approach you try does not need to work well.

2.3 Paper Specifications

The deliverable for this assignment will be a short (3 pages, single spaced including figures) paper that describes what you tried, why you tried it and what you learned from it. In particular, your paper should address the following questions.

1. Describe your approach to creating an assembler. Please be as specific as possible in describing your approach and why/how you chose this particular approach. You may also include information about things you initially tried, but that didn't work before you tried something else.

2. What modification to your data structures for storing the De Bruijn graph were necessary? For example, maybe you had to change between storing multi-edges as separate edges to just a single edge with a multiplicity. Maybe you needed to add a new data structure that kept track of which edges corresponded to each read. Etc.

3. How well did your approach work (on both error-free data and data with errors)? **Use data to support your conclusions.** For example, inclusion of plots or figures (or specific N50, L50 or other scoring metrics) may be useful here. Can you explain why your approach either worked well or worked poorly? Conjectures are also reasonable here if you aren't sure why your approach performed the way it did. At a minimum, your analysis should include your N50 and L50 scores for different values of k for the two datasets you created in part 1 of this assignment (`sample_c12_r_50_e0.00.txt` and `sample_c12_r_50_e0.01.txt`).

4. What did you learn from this assignment? Please spend some time reflecting on what was interesting or challenging with this assignment and report back.

2.4 Optional Extensions

There are a number of additional features that you may add to your simulation and assembly programs. If you choose to implement extra features, please make sure to discuss them in your write-up. Here are a few ideas of extra features you could implement:

- Improving the visualizations created by your assembler can often times be useful. If you did the extra credit in HW5, you could modify the DOT file you created so that the DOT file of your De Bruijn graph highlights (with color) the paths in the graph that correspond to your assembled contigs. You could also make your program output a different type of visualization called a contig map where reads are "aligned" to their corresponding place in each contig. For instance, the output of such a file for a single contig `GTGGACCTAA` and 4 reads might look like the following. (This type of output really useful for debugging).

```
G T G G A C C T A A
G T G G
  T G G A C
      G A C C T A
          C C T A A
```

- Make your simulator and assembler handle double stranded genomes (so, reads may come from either strand).

- Anything else you think of that would be a good improvement.

3 What to hand in

Please hand in to Moodle a zip file containing all of the following:

1. Your short paper formatted as a PDF.

2. The Python scripts as described above, and any other code you wrote to make it all work.

3. A README that explains the following:

    - How to compile your code (if necessary) and any dependencies (i.e., Biopython).

- A description of any known bugs.
- Any major design decisions that may affect the output of your programs.
- The names of anyone you discussed the project with and any outside resources you consulted when writing the code.

If you are working in a partnership, only one partner needs to turn in the code, but make sure that both your names are clearly indicated in both the code and the accompanying paper.

4 Grading

- **Code Specifications (4 points)** - Does your code adhere to the exact specifications outlined above? This includes, does it compute N50 and L50 correctly? Do you include a README with the requested information?
- **Paper (16 points)** - Your paper will be graded according to the rubric on the following page.

	1 - Poor	2 - Fair	3 - Good	4 - Excellent
Assembler Description	Description provides a broad overview, but is lacking many key details or motivation behind why this approach was chosen.	Description provides a broad overview, but is either lacking some key details or the motivation behind why this approach was chosen.	Description provides a broad overview, but is lacking some minor details or motivation behind why this approach is not throughly or clearly described.	Description provides a broad overview and all key details behind the chosen approach. Motivation for why approach was chosen is clearly and thoroughly explained.
Data Structure Modifications	Description of modifications is vague and hard to understand, and does not contain explanations of why these modifications were necessary.	Description of modifications is understandable with some effort by the reader, but lacks clear explanation on why these modifications were necessary.	Description of modifications is generally understandable by a reader and for the most part is accompanied by clear explanations of why these modifications were needed, including the choice of data structures.	Description of modifications are easily understandable by a reader and are always accompanied by clear explanations of why these modifications were needed, including the specific choice of data structures.
Analysis of Performance	Analysis includes the requested N50 and L50 scores, but little else.	Analysis includes the requested N50 and L50 scores and a general description of performance on both error-free and error containing data, but lacks sufficient concrete evidence backing up these claims. Description of why the approach either worked or didn't work well is not included or lacks sufficient detail.	Analysis includes the requested N50 and L50 scores as well as conclusions about the performance of the approach on error-free and error containing data, backed up by evidence. Description of why the approach either worked or didn't work well lacks some details.	Analysis includes the requested N50 and L50 scores as well as conclusions about the performance of the approach on error-free and error containing data, backed up by a variety of evidences. Description of why the approach either worked or didn't work well is well reasoned, thorough, and clearly written.
Reflection	Analysis is shallow and does not present a detailed reflection on what was challenging or learned during the process.	Some reflection is evident, but analysis of either what was learned or what was challenging lacks some depth.	Moderate reflection is evident as analysis of both what was learned and what was challenging is included.	Evidence of deep reflection is apparent as analysis of both what was learned and what was challenging is included, and furthermore, makes key connections back to in class discussions or readings.