Please upload your solutions on Gradescope. You can use \LaTeX{} or a word document to write up your answers, but we prefer you use \LaTeX{}. You may scan hand-written work or images for parts of solutions only if they are extremely clean and legible. Please ensure that your name does not appear anywhere in your handin.

**Problem 1: Global alignment table**

It is the Winter Olympics and athletes from around the world are competing in Beijing, China! A computational biologist is so excited to attend and watch the skijoring competition (skijoring is a sport where an animal pulls a person on skis - this animal can be a dog or a horse). To prove how much they love the Winter Olympics and skijoring, the computational biologist decides to perform a global alignment on the two words!

Create and fill out the global alignment dynamic programming table, with scores and back pointers, to align \textsc{skijoring} and \textsc{beijing}. For scoring, use the match bonus $+1$, mismatch penalty $-1$, and indel penalty $-1$. When there is a tie between a mismatch and an indel alignment, use a back pointer that indicates a mismatch. What is the score of the optimal alignment and to which alignment does this score correspond?

**Problem 2: Multiple alignment**

In class you have learned about the global alignment of two strings. A natural question to ask is how we might go about aligning three strings. Your task is to design an efficient algorithm that solves the following problem.

### Global Alignment of Three Strings

**Input:** Strings $u$, $v$, and $w$ and a scoring function $\delta$ that takes in three arguments.

**Output:** An alignment of $u$, $v$, and $w$ for which the score is maximal (as defined by $\delta$) among all alignments of $u$, $v$, and $w$.

Whereas we had to work fairly hard to build up the original binary global alignment algorithm from scratch, it turns out that we can obtain an algorithm for the global alignment of three strings by making some careful modifications to the algorithm you’ve seen in class.
In your solution, you must

• describe how the structure of the dynamic programming table changes in the extension to three strings,
• give a new recurrence relation for finding the score of an optimal alignment of a given combination of prefixes of \( u, v, \) and \( w, \)
• specify the order in which the new dynamic programming table should be filled out.

If you wish, you may also include diagrams, pseudocode, mathematical expressions, and plain English text in your answer. However, please do not submit a block of code with no accompanying explanation.

Once we know how to extend our global alignment algorithm to three strings, we might consider making the general extension to \( k \) strings:

• Show that if we continue to insist on constructing dynamic programming tables, the runtime of our algorithm will be \( O((2n)^k) \) for \( k \) sequences of length \( n \). Since this approach is exponential in \( k \), we will quickly hit a computational wall in attempting to align sequences of even modest length.

**Bonus:** Sketch in a few sentences a reasonable heuristic we might use to skirt this problem if we would still like to align multiple sequences.

**Problem 3: Counting global alignments**

When implementing algorithms, there are different approaches to solving a problem. Brute force programming is a direct approach to solving a task that depends more on computer processing power and memory, while dynamic programming optimizes by breaking a task into smaller problems and drawing upon solutions to the smaller problems in order to solve the overall task.

A brute force approach to global alignment would iterate through every possible alignment of the two input strings, computing the score and storing the max along the way. To understand the performance of such an approach, we need to know the number of possible alignments between two strings, one of length \( m \) and the other of length \( n \). Finding an exact expression for the number of possible alignments is actually fairly difficult. In this problem, we simply ask you to determine whether the number of alignments is

1. polynomial in \( m \) and \( n \)
2. logarithmic in \( m \) and \( n \)
3. exponential in \( m \) and \( n \)

Indicate your answer choice by number and justify your response.

**Problem 4: A Fowl Virus**

In order to feed the hungry Olympians this winter season, lots of chickens need to be raised. However, a farmer has noticed a disturbing trend in their new chicken farm: the chickens have started dying at alarming rates! They recruit you, a seasoned computational biologist, to get to the root of the problem. You
realize that there is a viral epidemic ravaging the farmer’s coops. Luckily, you happen to live next to a sequencing center. You manage to isolate some of the virus and sequence it. Now, with the sequence in your flash drive, it is up to you to save the chickens for the Olympians!

In the support directory (obtained by running `cs181_setup hw1`) you’ll find the sequence for the terrible virus plaguing the fowl. You know that the virus is a retrovirus and is inserting genes into the poor hens’ DNA. It’s up to you to find out what gene or genes are causing this farmhouse mayhem. Fortunately, you know about BLAST (Basic Local Alignment Search Tool). Navigate to the BLAST website. You’re only interested in your chicken’s genome so type “chickens” into the species search box and click “search.” Now, to find out what gene the virus is infecting your chickens with, copy and paste the viral genome into the box asking for a FASTA sequence and click “BLAST.” It will take a short time (1-2 minutes) before NCBI returns your query. Once it does, investigate the alignments it returns and try to get to the bottom of this plague.

**Task:** Figure out what gene your chickens are being infected with and what disease is killing them as a result. Specifically, make sure you check out the alignment for chromosome 20!

**Hint:** Clicking the “CDS Feature” checkbox in the “Alignments” tab will list genes or proteins coded within each range of the alignment, and selecting “GenBank” for any particular range will provide you with more information about the features found there. If you’re not sure whether a feature is significant, try looking up some of the individual words in that feature!

Be sure to provide justification for your answers. The virus itself has an interesting history; if you’re interested, try to figure out what virus it is exactly.

**Bonus:** The general structure of a retrovirus genome is composed of coding regions for the gag-pol-env polypeptides (NOT proteins) and other proteins that assemble the polypeptides into proteins. Using this information, find a closely related virus for the virus above. Give a short description of how you found the related protein.

**Problem 5: NCBI**

COVID-19, officially declared a “pandemic” in March 2020, is caused by a strain of coronavirus called SARS-CoV-2. SARS-CoV-2, like other coronaviruses, contains four main structural proteins: the spike (S), envelope (E), and membrane (M) proteins, which make up the viral envelope, and the nucleocapsid (N) protein, which contains the genetic material.

In your first programming project, you will work with sequence data to determine whether a specific SARS-CoV-2 protein is present in patient samples. The goal of this problem is to acquire such sequence data.

The primary resource for biomedical data is the collection of databases housed by the National Center for Biotechnology Information (NCBI). Navigate to the [NCBI homepage](https://www.ncbi.nlm.nih.gov) in your browser. Enter “SARS CoV-2” in the search box and hit enter. The NCBI’s user interface is a little dated, but it is also rich in
information.

Click on the link to the reference genome sequence. A reference genome (or reference assembly) is a digital nucleic acid sequence designed to serve as a representative example of a species’ DNA content. Reference genomes are often assembled by sequencing the DNA from a number of samples, and thus do not necessarily represent the genome of any single organism. The Human Genome Project took 13 years to assemble the first complete human genome. Today, it takes just over 24 hours to fully sequence a person’s DNA. The speedup is partly due to computational and technological advances over the last two decades, but it is also because we have a reference genome to serve as a guide. Sequencing DNA using a reference genome is like putting together a puzzle using the picture on the front of the box.

Returning to the coronavirus genome, scroll down to the feature containing the “surface glycoprotein“.

**Task:** Determine which of the four structural SARS-CoV-2 protein sequences this protein corresponds to.

We are interested in this protein sequence, so click on the “protein ID”. On the top of the “surface glycoprotein” page, click on the “FASTA” link. Download its corresponding amino acid sequence by selecting “Send to” on the top right. Choose “File” as the destination and FASTA as the format. FASTA is a standardized file format for biological sequence data.

**Task:** Turn this sequence in along with your homework.

Now navigate back to the original coronavirus complete reference genome. For the surface glycoprotein feature of the coronavirus genome, you should see two sections: a gene section and a CDS section. CDS stands for “coding sequence,” i.e. the part of the gene that is transcribed to mRNA. You just downloaded the translated amino acid sequence of this protein.

**Task:** Determine the name of the gene that encodes this protein, listed in the corresponding gene section.

Let’s do one more search on NCBI to determine find more information about this gene that encodes the surface glycoprotein of SARS-CoV-2. Navigate back to the search bar at the top of the page. Select the “Gene” option in the database dropdown menu and then type in the name of the gene you just identified. Click on the appropriate gene for the surface glycoprotein.

This page provides a summary of the gene functionality, mentioning its importance in vaccine development, antibody therapies, and diagnostic antigen-based tests. Let’s learn a bit more about the structure of this gene and the protein it encodes.

**Task:** List the four conserved domains identified in the genome annotation section. Domains are distinct functional and/or structural units of proteins. Conserved domains are the recurring units found in molecular evolution, identified by conserved sequence patterns or motifs.

Lastly, click on each of these conserved domains. This takes you to a page detailing the conserved protein
domain family, including a brief description of the corresponding function of the domain family.

**Task:** What is the function of these conserved domains?

That’s it, you’re done! You have successfully downloaded a real protein sequence from the internet and navigated NCBI. The fun part is to actually run algorithms on the data and interpret the biological significance of the results. You’ll be doing just that on your first project.

**Problem 6: DNA Sequencing**

DNA sequencing has become indispensable to biological research, enabling us to inspect genomes like those you’ll see in this class. With constant innovation, this technology has become more readily available, cheaper, and quicker. There are various companies that provide sequencing technology—arguably the most prevalent is Illumina:

"Illumina is a leading developer, manufacturer, and marketer of life science tools and integrated systems for large-scale analysis of genetic variation and function. These systems are enabling studies that were not even imaginable just a few years ago, and moving us closer to the realization of personalized medicine.”

Read Illumina’s [beginner’s guide](#) to next-generation sequencing. If you’re interested, watch [this](#) brief video on How NGS Works.

a. In your own words, explain the process of next-generation sequencing. Try to be as concise as possible!

b. Considering logistics and the everyday decisions of a researcher, give at least 3 specific examples of how human bias or error may impact sequencing output. Possible areas of focus include:

- Workflow Development
- Sample Collection
- Library Preparation
- Sequencing
- Data Analysis
- Handling and Communicating Results

**Problem 7: Reference Genomes**

Read [this](#) article about the impact of the human reference genome on personalized medicine.

a. In the construction of the human reference genome, what groups of individuals were represented in the process and what groups of individuals were not represented? How could this affect the results of future studies that rely on this reference genome?
Problem 8: Introduction to Ethics Issues

Read through this list of policy issues in genomics.

a. What are some issues that youâd like to explore further? Are there any other issues or specific topics that you think might be interesting?

b. How may some of these issues apply to your future work, if applicable?