**OPTIONAL – If not using the homework assignment/term project**

Here are some proposed exam/quiz questions to ask. Questions 4 & 5 can be used as an assignment for elaboration and/or evaluation. I do not recommend asking Questions 4 or 5 on a timed exam.

*Multiple Choice Question*

1. What is pseudocode?
    a. A formal programming language used to write complex algorithms.
    b. A broad description of the main steps of the algorithm.
    c. A high-level description of an algorithm using plain language and informal notation.
    d. A type of code used to encrypt sensitive information.

    Answer: C

    (I have to add a LO. "SWBAT **define** pseudocode and its uses") (Level 1)

*Select All that Apply Question*

2. What are some qualities of "good" pseudocode? Select all that apply.
    a. Specific to one programming language
    b. Defines the program input and output, including variable names and data structures
    c. Breaks down project into small, manageable pieces.
    d. Uses technical language
    e. Variable definitions include intended data type
    f. Indentation is not required

    Answers: B, C, E

    (Relates to LO starting with "describe…") (Level 2)

*Short Answer Question*

3. Why is pseudocode used as the first step in algorithm development? List 2 reasons.

    Answer: Many correct answers. Looking for any key phrases like the following:
    - Easier to team code because the parts can be coded individually and are easily put together
    - Defines where the program starts, the process it completes, and the output
    - Easy transition from one language to another
    - Helps to simplify steps (avoids redundancy)
    - Easier to change your plan in plain language than in a formal programming language

    (Also relates to my new LO, but at level 2/3)

*Essay Questions (These questions are too long for students to successfully consider and read during an in-class exam. But you can turn them into a worksheet or a take home exam!)*

4. Write "good" pseudocode for a 2-person team to accomplish the following task:

   **Create a stacked bar chart of the amino acid composition of 5 proteins selected by the user.**

   YOUR INPUT: The user will provide a FASTA file with the protein ID and protein sequence.

   YOUR OUTPUT: There should be 2 charts as output.

   1. A stacked bar chart with individual amino acid side chain prevalence
   2. A stacked bar chart to visualize "classes" of amino acid side chains (polar, nonpolar, charged)

   *Helpful considerations:*

   - There are two ways to count all amino acids of each type:
     - Read the protein chain one-by-one and add to the counter for each amino acid (counter can be a dictionary or data frame).
     - Sort the string alphabetically, cut it into smaller strings each containing one letter, take the length of the smaller string.
   - You can add up the total counts of each amino acid, side chain type either while you are counting, or after all counting is done.

Answer: All their pseudocode will be formatted differently (sad for grading, but helpful for assessing their ability to do this). The following is not pseudocode, but an outline of the steps their pseudocode should cover. Students will opt to "hand off" coding responsibilities at different points. Any point is allowable if they write the output of the first as the input of the second. The output should include detailed information about the structure and type of data being handed off.

**Step 1: Input**

Prompt the user to provide a FASTA file containing protein IDs and sequences.

**Step 2: Initialize Counters**

Initialize two dictionaries/data frames:

amino_acid_counts to store individual amino acid counts.

group_counts to store counts of amino acid groups (polar, nonpolar, charged).

**Step 3: Define Amino Acid Groups (Anywhere before step 6)**

Create a dictionary/data frame (amino_acid_groups) to classify amino acids into groups (e.g., polar, nonpolar, charged).

**Step 4: Read FASTA File (can also be Step 2)**

Read the FASTA file, extracting protein IDs and sequences.

Store protein sequences in a dictionary with IDs as keys (or a specific data frame structure telling me what is the row/column names and where the sequences and IDs are stored).

**Step 5: Amino Acid Counting**

For each selected protein, iterate through its amino acids in the sequence:

Increment the corresponding amino acid count in amino_acid_counts.

Determine the amino acid group and increment the corresponding group count in group_counts.

**Step 6: Create Stacked Bar Charts**

Create two stacked bar charts:

Individual Amino Acid Composition:
X-axis: Amino acids (A, R, N, ...).
Y-axis: Count.
Title: "Amino Acid Composition."
Amino Acid Group Composition:
X-axis: Amino acid groups (polar, nonpolar, charged).
Y-axis: Count.
Title: "Amino Acid Group Composition."

**Step 8: Display or Save Charts**

Display the stacked bar charts to the user or save them to a file for later reference.

(Aligns with LO beginning with "plan…") (Level 6)

5. The following pseudocode is designed to count the number of each type of nucleotide in a gene from a FASTA file and output as a summary in the command line.

```
# Step 1: Input RNA Sequence
Get FASTA file from user
Read rna_sequence # a string

# Step 2: Data Validation
Make sure that it only has "A", "U", "C", & "G"

# Step 3: Initialize Count Variables
count_A = 0 #all are integers
count_U = 0
count_C = 0
count_G = 0

# Step 4: Count Nucleotides
Add to counters

# Step 5: Calculate Nucleotide Composition
percentage_A = (count_A / total_nucleotides) * 100 #a float, limited to 2 decimal places
percentage_U = (count_U / total_nucleotides) * 100
percentage_C = (count_C / total_nucleotides) * 100
percentage_G = (count_G / total_nucleotides) * 100

# Step 6: Output Summary
Display "Total Nucleotides:", total_nucleotides
Display "A:", count_A, "(", percentage_A, "% )"
Display "U:", count_U, "(", percentage_U, "% )"
```

```
    Display "C:", count_C, "(", percentage_C, "% )"
    Display "G:", count_G, "(", percentage_G, "% )"

    OUTPUT: The output is displayed in the terminal (command line) as text.
```

a. Critique the pseudocode. What is written well? List these items. (Level 4/5)
b. Critique the pseudocode. What needs improvement? List these items. (Level 4/5)
c. Add/rewrite sections of pseudocode you identified in part b to improve the pseudocode. The final pseudocode should have all the aspects of "good" pseudocode. (Level 6)

Answer:

a. This is done well:
   - Steps are clearly defined and do not perform more than one function.
   - Specifies data types for variables.
   - Has "OUTPUT" for the program.
   - Program does not use programming language specific writing.

b. This needs improvement:
   - Missing "INPUT" for the program.
   - Program does not do anything (exit the program/allow user to edit input) if the sequence contains wrong letters.
   - Step 4 is not detailed enough.

c. Answers will vary. Students can provide corrections in many forms. This is just one example. The areas highlighted are the sections that students should have identified as "needing improvement".

```
INPUT: FASTA file containing an mRNA gene sequence
# Step 1: Input RNA Sequence
Prompt user to enter the RNA sequence
Read rna_sequence # a string

# Step 2: Data Validation
if rna_sequence contains any characters not in "AUCGaucg":
    Display "Invalid DNA sequence. Please enter a valid DNA sequence."
    Exit the program

# Step 3: Initialize Count Variables
count_A = 0 #All are integers
count_U = 0
count_C = 0
count_G = 0

# Step 4: Count Nucleotides
total_nucleotides = length of dna_sequence
For each character in dna_sequence:
    If the character is 'A' or 'a':
        Increment count_A
    Else if the character is 'U' or 'U':
        Increment count_T
    Else if the character is 'C' or 'c':
        Increment count_C
    Else if the character is 'G' or 'g':
        Increment count_G

# Step 5: Calculate Nucleotide Composition
          percentage_A = (count_A / total_nucleotides) * 100 #a float, limited to 2 decimal places
percentage_U = (count_U / total_nucleotides) * 100
percentage_C = (count_C / total_nucleotides) * 100
percentage_G = (count_G / total_nucleotides) * 100
# Step 6: Output Summary
Display "Total Nucleotides:", total_nucleotides
Display "A:", count_A, "(", percentage_A, "% )"
Display "U:", count_U, "(", percentage_U, "% )"
Display "C:", count_C, "(", percentage_C, "% )"
Display "G:", count_G, "(", percentage_G, "% )"
```

OUTPUT: The output is displayed in the terminal as text.

(Aligns with LOs beginning with "modify…" and "critique…") (Levels 4-6)