import sys  # Import sys to handle command-line arguments

# Codon to amino acid translation table

codon\_table = {

    'ATA':'I', 'ATC':'I', 'ATT':'I', 'ATG':'M',  # Isoleucine and Methionine

    'ACA':'T', 'ACC':'T', 'ACG':'T', 'ACT':'T',  # Threonine

    'AAC':'N', 'AAT':'N', 'AAA':'K', 'AAG':'K',  # Asparagine, Lysine

    'AGC':'S', 'AGT':'S', 'AGA':'R', 'AGG':'R',  # Serine, Arginine

    'CTA':'L', 'CTC':'L', 'CTG':'L', 'CTT':'L',  # Leucine

    'CCA':'P', 'CCC':'P', 'CCG':'P', 'CCT':'P',  # Proline

    'CAC':'H', 'CAT':'H', 'CAA':'Q', 'CAG':'Q',  # Histidine, Glutamine

    'CGA':'R', 'CGC':'R', 'CGG':'R', 'CGT':'R',  # Arginine

    'GTA':'V', 'GTC':'V', 'GTG':'V', 'GTT':'V',  # Valine

    'GCA':'A', 'GCC':'A', 'GCG':'A', 'GCT':'A',  # Alanine

    'GAC':'D', 'GAT':'D', 'GAA':'E', 'GAG':'E',  # Aspartic acid, Glutamic acid

    'GGA':'G', 'GGC':'G', 'GGG':'G', 'GGT':'G',  # Glycine

    'TCA':'S', 'TCC':'S', 'TCG':'S', 'TCT':'S',  # Serine

    'TTC':'F', 'TTT':'F', 'TTA':'L', 'TTG':'L',  # Phenylalanine, Leucine

    'TAC':'Y', 'TAT':'Y',                        # Tyrosine

    'TGC':'C', 'TGT':'C',                        # Cysteine

    'TGG':'W',                                   # Tryptophan

    'TAA':'', 'TAG':'', 'TGA':''                 # Stop codons (empty strings as placeholders)

}

def read\_fna(filename):

    """Read the DNA sequence from a .fna file, removing header lines."""

    with open(filename, 'r') as file:

        lines = file.readlines()

    # Join lines together, excluding header (lines starting with '>')

    sequence = ''.join([line.strip() for line in lines if not line.startswith('>')])

    return sequence

def reverse\_complement(sequence):

    """Generate the reverse complement of a DNA sequence, handling unknown nucleotides."""

    complement = {'A': 'T', 'T': 'A', 'G': 'C', 'C': 'G', 'N': 'N'}  # Handle 'N' by mapping it to 'N'

    return ''.join(complement.get(base, 'N') for base in reversed(sequence))  # Use 'N' as a fallback for unknown bases

def translate\_sequence(dna\_seq):

    """Translate a DNA sequence into an amino acid sequence."""

    amino\_acids = []

    for i in range(0, len(dna\_seq) - 2, 3):  # Process in triplets (codons)

        codon = dna\_seq[i:i+3]

        amino\_acids.append(codon\_table.get(codon, '?'))  # Translate codon, '?' for unknown codons

    return ''.join(amino\_acids)

def find\_genes\_six\_frames(sequence):

    """Find potential genes in all six reading frames (three forward, three reverse)."""

    start\_codon = 'ATG'

    stop\_codons = ['TAA', 'TAG', 'TGA']

    genes = []

    # Search in all three forward frames

    for frame in range(3):

        for i in range(frame, len(sequence) - 2, 3):

            codon = sequence[i:i+3]

            if codon == start\_codon:

                for j in range(i + 3, len(sequence) - 2, 3):

                    stop\_codon = sequence[j:j+3]

                    if stop\_codon in stop\_codons:

                        gene\_seq = sequence[i:j+3]  # Extract gene from start to stop codon

                        amino\_seq = translate\_sequence(gene\_seq)  # Translate gene sequence

                        genes.append(('Forward', frame + 1, i + 1, j + 3, gene\_seq, amino\_seq))

                        break  # Stop after finding the first valid gene in this frame

    reverse\_seq = reverse\_complement(sequence)  # Get reverse complement of the sequence

    # Search in all three reverse frames

    for frame in range(3):

        for i in range(frame, len(reverse\_seq) - 2, 3):

            codon = reverse\_seq[i:i+3]

            if codon == start\_codon:

                for j in range(i + 3, len(reverse\_seq) - 2, 3):

                    stop\_codon = reverse\_seq[j:j+3]

                    if stop\_codon in stop\_codons:

                        start\_pos = len(sequence) - (i + 1)  # Convert reverse position to original sequence position

                        stop\_pos = len(sequence) - (j + 3)

                        gene\_seq = reverse\_seq[i:j+3]  # Extract gene sequence

                        amino\_seq = translate\_sequence(gene\_seq)  # Translate gene sequence

                        genes.append(('Reverse', frame + 1, stop\_pos + 1, start\_pos + 1, gene\_seq, amino\_seq))

                        break  # Stop after finding the first valid gene in this frame

    return genes

def main\_task\_4(filename):

    """Main function to read the sequence and find genes in six reading frames."""

    sequence = read\_fna(filename)  # Read sequence from file

    genes = find\_genes\_six\_frames(sequence)  # Find genes in all six frames

    if genes:

        for gene in genes:

            strand, frame, start, stop, gene\_seq, amino\_seq = gene

            # Output the results with gene details

            print(f"{strand} Strand Frame {frame}: Start at {start}, Stop at {stop}")

            print(f"DNA Sequence: {gene\_seq}")

            print(f"Amino Acid Sequence: {amino\_seq}\n")

    else:

        print("No valid start-stop codon region found.")

if \_\_name\_\_ == "\_\_main\_\_":

    fna\_file = sys.argv[1]  # Take the filename from command-line arguments

    main\_task\_4(fna\_file)  # Execute main function with the input file