Problem #1:

def nt\_count(s):

# Count occurrences of each nucleotide

count\_a = s.count('A')

count\_c = s.count('C')

count\_g = s.count('G')

count\_t = s.count('T')

# Return the counts as a formatted string

return f"{count\_a} {count\_c} {count\_g} {count\_t}"

# Sample dataset

s = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC"

# Get the result

result = nt\_count(s)

# Print the result

print(result)

def nt\_count(s):

# Count occurrences of each nucleotide

count\_a = 0

count\_c = 0

count\_g = 0

count\_t = 0

for nt in s:

if nt == "A":

count\_a += 1

elif nt == "C":

count\_c += 1

elif nt == "G":

count\_g += 1

elif nt == "T":

count\_t += 1

# Return the counts as a formatted string

return f"{count\_a} {count\_c} {count\_g} {count\_t}"

# Sample dataset

s = "AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC"

# Get the result

result = nt\_count(s)

# Print the result

print(result)

Problem # 2:

def dna\_to\_rna(t):

for nt in t:

if nt == "T":

"".join("U")

else:

nt = "".join(nt)

return t

def dna\_to\_rna(t):

u = t.replace("T", "U")

return u

# Sample dataset

t = "GATGGAACTTGACTACGTAAATT"

# Get the result

result = dna\_to\_rna(t)

# Print the result

print(result)

Problem # 4:

def rabbit\_pairs(n, k):

if n == 1 or n == 2:

return 1

rabbits = [0] \* n

rabbits[0] = 1

rabbits[1] = 1

for i in range(2, n):

rabbits[i] = rabbits[i-1] + k \* rabbits[i-2]

return rabbits[-1]

# Sample dataset

n = 36

k = 4

# Get the result

result = rabbit\_pairs(n, k)

# Print the result

print(result)

Problem #6:

def hamming\_distance(s, t):

nt\_difference = 0

for nt in range(len(s)):

if s[nt] != t[nt]:

nt\_difference += 1

return nt\_difference

s = "GAGCCTACTAACGGGAT"

t = "CATCGTAATGACGGCCT"

result = hamming\_distance(s, t)

print(result)

Problem #7:

Problem #8:

def rna\_to\_protein(s):

protein = ""

rna\_codon\_table = {

"UUU": "F", "CUU": "L", "AUU": "I", "GUU": "V",

"UUC": "F", "CUC": "L", "AUC": "I", "GUC": "V",

"UUA": "L", "CUA": "L", "AUA": "I", "GUA": "V",

"UUG": "L", "CUG": "L", "AUG": "M", "GUG": "V",

"UCU": "S", "CCU": "P", "ACU": "T", "GCU": "A",

"UCC": "S", "CCC": "P", "ACC": "T", "GCC": "A",

"UCA": "S", "CCA": "P", "ACA": "T", "GCA": "A",

"UCG": "S", "CCG": "P", "ACG": "T", "GCG": "A",

"UAU": "Y", "CAU": "H", "AAU": "N", "GAU": "D",

"UAC": "Y", "CAC": "H", "AAC": "N", "GAC": "D",

"UAA": "Stop", "CAA": "Q", "AAA": "K", "GAA": "E",

"UAG": "Stop", "CAG": "Q", "AAG": "K", "GAG": "E",

"UGU": "C", "CGU": "R", "AGU": "S", "GGU": "G",

"UGC": "C", "CGC": "R", "AGC": "S", "GGC": "G",

"UGA": "Stop", "CGA": "R", "AGA": "R", "GGA": "G",

"UGG": "W", "CGG": "R", "AGG": "R", "GGG": "G",

}

for i in range(0, len(s), 3):

codon = s[i:i+3]

if codon in rna\_codon\_table:

amino\_acid = rna\_codon\_table[codon]

if amino\_acid == "Stop":

break

protein += amino\_acid

return protein

s = "AUGGCCAUGGCGCCCAGAACUGAGAUCAAUAGUACCCGUAUUAACGGGUGA"

result = rna\_to\_protein(s)

print(result)

Problem #9:

def DNA\_motif(s, t):

locations = []

for nt in range(len(s) - len(t) + 1):

if s[nt:nt + len(t)] == t:

locations.append(nt + 1)

return locations

s = "GATATATGCATATACTT"

t = "ATAT"

result = DNA\_motif(s, t)

print(" ".join(map(str, result)))

Problem #20:

def protein\_weight(P):

weight = 0

amino\_acid\_masses = {

"A": 71.03711,

"C": 103.00919,

"D": 115.02694,

"E": 129.04259,

"F": 147.06841,

"G": 57.02146,

"H": 137.05891,

"I": 113.08406,

"K": 128.09496,

"L": 113.08406,

"M": 131.04049,

"N": 114.04293,

"P": 97.05276,

"Q": 128.05858,

"R": 156.10111,

"S": 87.03203,

"T": 101.04768,

"V": 99.06841,

"W": 186.07931,

"Y": 163.06333

}

for amino\_acid in P:

if amino\_acid in amino\_acid\_masses:

weight += amino\_acid\_masses[amino\_acid]

return weight

P = "SKADYEK"

result = protein\_weight(P)

print(result)