Practical 7

In each of the questions, after the output has been generated, an optimal code has been used to beautifully convert the generated code into a neatly tabulated column in pandas, where multi-column scenarios are also possible. The code to do so is given below:

```
import pandas as pd
import matplotlib.pyplot as plt

# The columns with multi-column possibilities
# Contains the individual column and group column names depending on
    situation
col =
["Amino Acid", "Sequence 1, Composition in %",
 "Sequence 2, Composition in %", "Sequence 3, Composition in %"]

# You put the lists for respective columns in the dictionary data
data =
{"Amino Acid": amino_acid, "Sequence 1, Composition in %": amino_acid_comp1,
 "Sequence 2, Composition in %": amino_acid_comp2,
 "Sequence 3, Composition in %":amino_acid_comp3}

# Converting the above into a dataframe and establishing multi-column
    scenarios
df = pd.DataFrame(data, columns=col)

# Grouping together the columns that have same root group
a = df.columns.str.split(', ', expand=True).values
df.columns = pd.MultiIndex.from_tuples([('', x[0]) if pd.isnull(x[1])
    else x for x in a])

# Printing the df
df
```

Listing 1. Code to generate a neat tabulated column

Now, each of the following questions requiring code will contain the function that is essential to obtain the objective of the given question, and the output obtained thus by using code in a similar manner as above.

Question 1. Compute the amino acid composition of the sequences given below. Provide the output as a table of amino acid percentage values for each sequence, and comment on the results.

Human mitochondrial beta barrel membrane protein VDAC1

1.
RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATA
TPEYLAALKQKSRHAA
2.
AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPM
IAVGLGLYVMFAVA
3.
AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYW
EGGDEGAGKHSLSFAPVFVYEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSL
NFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQPNDGIESYSLFYKIPI

Solution. The code to compute the amino acid composition for a given sequence is given below:

```
import pandas as pd
import matplotlib.pyplot as plt

amino_acid_list =
{"A": 0,"R": 1,"N": 2,"D": 3,"C": 4,"Q": 5,"E": 6,"G": 7,"H": 8,"I": 9,
 "L":10,"K":11,"M":12,"F":13,"P":14,"S":15,"T":16,"W":17,"Y":18,"V":19}

# The below list is just for displaying in the output
 amino_acid =
 ["A", "R", "N", "D", "C", "Q", "E", "G", "H", "I",
  "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"]

# Computing the composition of the sequence
def composition(seq):
    amino_acid_count = [0 for i in range(20)] # 20 amino acids
    for i in range(len(seq)):
        amino_acid_count[amino_acid_list[seq[i]]] += 1

    len_seq = len(seq)
    for i in range(len(amino_acid_count)):
        amino_acid_count[i] /= len_seq
        amino_acid_count[i] *= 100

    return amino_acid_count

seq1 =
"RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA"
seq2 =
"AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA"
seq3 =
"AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFV
 YEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQP
 NDGIESYSLFYKIPI"

amino_acid_comp1 = composition(seq1)
data = {"Amino Acid": amino_acid, "Composition in %": amino_acid_comp1}
df1 = pd.DataFrame(data)

amino_acid_comp2 = composition(seq2)
data = {"Amino Acid": amino_acid, "Composition in %": amino_acid_comp2}
df2 = pd.DataFrame(data)

amino_acid_comp3 = composition(seq3)
data = {"Amino Acid": amino_acid, "Composition in %": amino_acid_comp3}
df3 = pd.DataFrame(data)
```

LISTING 2. Code to compute the amino acid composition

Here are some comments on the amino acid compositions for the given sequences:

- Each sequence exhibits a unique distribution of amino acids. For instance, Seq2 has a notably higher proportion of isoleucine (I) compared to Seq1 and Seq3, indicating a potential sequence-specific preference for this amino acid.
- The sequences vary in the diversity of amino acids present. For example, Seq1 and Seq3 show relatively higher diversity with a broader range of amino acids represented compared to Seq2, which appears to have a more focused amino acid composition.

- Seq2 has relatively high percentages of leucine and isoleucine, indicating that it may have a higher representation of hydrophobic residues compared to the other two.
- Seq1 has a small percentage of cysteine, while Seq2 and Seq3 have none. This indicates that cysteine residues are less prevalent in the given three sequences, suggesting a potential absence of disulfide bonds.
- The composition of Alanine is sufficiently large in all three given sequences. Hence, Alanine plays an important role in each of the three sequences.

	Sequence 1	Sequence 2	Sequence 3
Amino Acid	Composition in %	Composition in %	Composition in %
A	17.910448	19.117647	10.596026
R	5.970149	2.941176	3.311258
N	1.492537	0.000000	3.311258
D	2.985075	2.941176	5.960265
C	1.492537	0.000000	0.000000
Q	1.492537	2.941176	3.311258
E	2.985075	1.470588	3.973510
G	5.970149	14.705882	15.231788
H	4.477612	0.000000	1.324503
I	1.492537	11.764706	5.298013
L	8.955224	13.235294	5.960265
K	7.462687	1.470588	3.973510
M	0.000000	7.352941	0.662252
F	1.492537	5.882353	5.298013
P	7.462687	4.411765	2.649007
S	5.970149	0.000000	9.933775
T	10.447761	1.470588	4.635762
W	5.970149	0.000000	3.973510
Y	4.477612	1.470588	5.298013
V	1.492537	8.823529	5.298013

FIGURE 1. The percentage composition of amino acids in the 3 sequences

Question 2. Assume the molecular weights of the 20 amino acid residues as given below. Compute the molecular weight of the three sequences given in question 1.

Residue	Mol.Wt	Residue	Mol.Wt	Residue	Mol.Wt	Residue	Mol.Wt
Ala	85	Cys	115	Asp	130	Glu	145
Phe	160	Gly	70	Trp	200	His	150
Ile	125	Lys	145	Leu	125	Met	143
Asn	130	Tyr	175	Pro	110	Gln	140
Arg	170	Ser	100	Thr	115	Val	110

Solution. Below is the code for the program to compute the mol. weight of the three sequences.

```
# Initializing the weights
mol_weights =
{"A":  85, "C": 115, "D": 130, "E": 145, "F": 160,
 "G":  70, "W": 200, "H": 150, "I": 125, "K": 145,
 "L": 125, "M": 143, "N": 130, "Y": 175, "P": 110,
 "Q": 140, "R": 170, "S": 100, "T": 115, "V": 110}

def molecular_weight(seq):
    weight = 0
    for i in range(len(seq)):
        weight += mol_weights[seq[i]]

    # It is important to subtract (N-1) * weight of water
    # When protein sequence forms, for every two amino acids forming a
   peptide bond, one molecule of water is lost.
    weight_water = 18
    weight -= (len(seq)-1) * weight_water

    return weight

seq1 =
"RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA"
seq2 =
"AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA"
seq3 =
"AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFV
 YEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQP
 NDGIESYSLFYKIPI"

w1 = molecular_weight(seq1)
w2 = molecular_weight(seq2)
w3 = molecular_weight(seq3)

print("Molecular weight of Sequence 1 is", w1)
print("Molecular weight of Sequence 2 is", w2)
print("Molecular weight of Sequence 3 is", w3)
```

Listing 3. Code to compute alignment score

The molecular weights of the three sequences are:

Sequence Number	Molecular weight of sequence
Sequence 1	7127
Sequence 2	6529
Sequence 3	15453

Question 3. The amino acid composition of a standard set of Group A (first value) and Group B (second value) proteins are given below. Identify whether the given sequences in Question 1 belong to Group A or Group B and write your answer.

Residue	A, B	Residue	A, B	Residue	A, B	Residue	A, B
Ala	8.47, 8.95	Cys	1.39, 0.47	Asp	5.97, 5.91	Glu	6.32, 4.78
Phe	3.91, 3.68	Gly	7.82, 8.54	Trp	1.44, 1.24	His	2.26, 1.25
Ile	5.71, 4.77	Lys	5.76, 4.93	Leu	8.48, 8.78	Met	2.21, 1.56
Asn	4.54, 5.74	Tyr	3.58, 4.13	Pro	4.63, 3.74	Gln	3.82, 4.75
Arg	4.93, 5.24	Ser	5.94, 8.05	Thr	5.79, 6.54	Val	7.02, 6.76

Solution. Below is the code to check whether the given sequences in Question 1 belong to the Group A or Group B based on the amino acid composition.

```
amino_acid_list =
{"A":  0, "R":  1, "N":  2, "D":  3, "C":  4,
 "Q":  5, "E":  6, "G":  7, "H":  8, "I":  9,
 "L": 10, "K": 11, "M": 12, "F": 13, "P": 14,
 "S": 15, "T": 16, "W": 17, "Y": 18, "V": 19}

amino_acid_index =
{ "0": "A", "1": "R", "2": "N", "3": "D", "4": "C",
  "5": "Q", "6": "E", "7": "G", "8": "H", "9": "I",
 "10": "L","11": "K","12": "M","13": "F","14": "P",
 "15": "S","16": "T","17": "W","18": "Y","19": "V"}

group_a =
{"A": 8.47, "D": 5.97, "C": 1.39, "E": 6.32, "T": 5.79,
 "F": 3.91, "G": 7.82, "H": 2.26, "I": 5.71, "V": 7.02,
 "K": 5.76, "L": 8.48, "M": 2.21, "N": 4.54, "W": 1.44,
 "P": 4.63, "Q": 3.82, "R": 4.93, "S": 5.94, "Y": 3.58}

group_b =
{"A": 8.95, "D": 5.91, "C": 0.47, "E": 4.78, "T": 6.54,
 "F": 3.68, "G": 8.54, "H": 1.25, "I": 4.77, "V": 6.76,
 "K": 4.93, "L": 8.78, "M": 1.56, "N": 5.74, "W": 1.24,
 "P": 3.74, "Q": 4.75, "R": 5.24, "S": 8.05, "Y": 4.13}

# I will use absolute error to measure the closeness
def closeness(seq):
    comp = composition(seq)
    for i in range(len(comp)):
        comp[i] *= 100
    close_A = []
    close_B = []
    equalAB = []
    dist_A = 0
    dist_B = 0

    for i in range(len(comp)):
        mod_diff_A = abs(group_a[amino_acid_index[str(i)]] - comp[i])
        mod_diff_B = abs(group_b[amino_acid_index[str(i)]] - comp[i])
        dist_A += mod_diff_A
        dist_B += mod_diff_B
        if mod_diff_A > mod_diff_B:
            close_B.append(amino_acid_index[str(i)])
        elif mod_diff_A < mod_diff_B:
```

```
            close_A.append(amino_acid_index[str(i)])
        else:
            equalAB.append(amino_acid_index[str(i)])

    output = "A" if dist_A < dist_B else "B"

    return close_A, close_B, equalAB, output

seq1 =
"RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA"
seq2 =
"AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA"
seq3 =
"AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFV
 YEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQP
 NDGIESYSLFYKIPI"

A_close1 = ["-" for i in range(20)]
B_close1 = ["-" for i in range(20)]
AB_equal1 = ["-" for i in range(20)]
A_close2 = ["-" for i in range(20)]
B_close2 = ["-" for i in range(20)]
AB_equal2 = ["-" for i in range(20)]
A_close3 = ["-" for i in range(20)]
B_close3 = ["-" for i in range(20)]
AB_equal3 = ["-" for i in range(20)]

A_close1, B_close1, AB_equal1, out1 = closeness(seq1)
A_close2, B_close2, AB_equal2, out2 = closeness(seq2)
A_close3, B_close3, AB_equal3, out3 = closeness(seq3)

print(out1)
print(out2)
print(out3)

data = {"close to A": A_close1, "close to B": B_close1, "equal to AB":
    AB_equal1}
df1 = pd.DataFrame(data, columns=["close to A", "close to B", "equal to AB"])

data = {"close to A": A_close2, "close to B": B_close2, "equal to AB":
    AB_equal2}
df2 = pd.DataFrame(data, columns=["close to A", "close to B", "equal to AB"])

data = {"close to A": A_close3, "close to B": B_close3, "equal to AB":
    AB_equal3}
df3 = pd.DataFrame(data, columns=["close to A", "close to B", "equal to AB"])
```

LISTING 4. Code to check closeness to Group A or Group B

I have printed the output as to whether it belongs to Group A or Group B in the code. The output is:

Sequence Number	Closest to which group
Sequence 1	Group A
Sequence 2	Group A
Sequence 3	Group B

I have also tabulated the closeness of each amino acid to Group A or Group B in each of the three sequences. The table is seen below:

Sequence 1			Sequence 2			Sequence 3		
Close to A	Close to B	Equal to AB	Close to A	Close to B	Equal to AB	Close to A	Close to B	Equal to AB
N	A	-	R	A	-	R	A	-
C	R	-	N	D	-	N	C	-
Q	D	-	Q	C	-	D	E	-
G	E	-	I	E	-	Q	G	-
H	I	-	M	G	-	I	H	-
K	L	-	F	H	-	L	K	-
P	M	-	P	L	-	F	M	-
S	F	-	S	K	-	T	P	-
W	T	-	T	W	-	W	S	-
-	Y	-	Y	-	-	-	Y	-
-	V	-	V	-	-	-	V	-

FIGURE 2. Each amino acid closeness to Groups A and B

Question 4. Compute the residue pair preference for the three sequences given in question 1. The required output is a 20x20 table showing the pair preferences

(1) $$(a)\frac{N_{ij} * 100}{N_i + N_j}$$

(2) $$(b)\frac{N_{ij} * 100}{N - 1}$$

(3) $$(c)\frac{N_{ij} * 100}{N_i * N_j}$$

List the top 10 preferred residues from each of the three pair-preferences.

Solution. The code to list top 10 preferred residues from each of the three pair-preferences is:

```
seq1 =
"RATPTRWPVGCFNRPWTKWSYDEALDGIKAAGYAWTGLLTASKPSLHHATATPEYLAALKQKSRHAA"
seq2 =
"AAAVMMGLAAIGAAIGIGILGGKFLEGAARQPDLIPLLRTQFFIVMGLVDAIPMIAVGLGLYVMFAVA"
seq3 =
"AADVSAAVGATGQSGMTYRLGLSWDWDKSWWQTSTGRLTGYWDAGYTYWEGGDEGAGKHSLSFAPVFV
 YEFAGDSIKPFIEAGIGVAAFSGTRVGDQNLGSSLNFEDRIGAGLKFANGQSVGVRAIHYSNAGLKQP
 NDGIESYSLFYKIPI"

amino_acid_list =
{"A":  0, "R":  1, "N":  2, "D":  3, "C":  4,
 "Q":  5, "E":  6, "G":  7, "H":  8, "I":  9,
 "L": 10, "K": 11, "M": 12, "F": 13, "P": 14,
 "S": 15, "T": 16, "W": 17, "Y": 18, "V": 19}
```

```
15
16 amino_acid_index =
17 { "0": "A", "1": "R", "2": "N", "3": "D", "4": "C",
18   "5": "Q", "6": "E", "7": "G", "8": "H", "9": "I",
19  "10": "L","11": "K","12": "M","13": "F","14": "P",
20  "15": "S","16": "T","17": "W","18": "Y","19": "V"}
21
22 def pair_pref(seq):
23     # Storing the pair-preference methods in different matrices
24     pair_matrix_a = [[0 for i in range(21)] for j in range(20)]
25     pair_matrix_b = [[0 for i in range(21)] for j in range(20)]
26     pair_matrix_c = [[0 for i in range(21)] for j in range(20)]
27     amino_freq = [0 for i in range(20)]
28
29     # Incrementing matrices on finding the pairs in sequences
30     for i in range(len(seq)):
31         amino_freq[amino_acid_list[seq[i]]] += 1
32         if i == len(seq) - 1:
33             continue
34         pair_matrix_a[amino_acid_list[seq[i]]][amino_acid_list[seq[i+1]]]+=1
35         pair_matrix_b[amino_acid_list[seq[i]]][amino_acid_list[seq[i+1]]]+=1
36         pair_matrix_c[amino_acid_list[seq[i]]][amino_acid_list[seq[i+1]]]+=1
37
38     #  (a) [Nij*100/(Ni+Nj)]
39     for i in range(20):
40         for j in range(20):
41             if pair_matrix_a[i][j] != 0:
42                 pair_matrix_a[i][j] *= 100
43                 pair_matrix_a[i][j] /= (amino_freq[i] + amino_freq[j])
44                 pair_matrix_a[i][j] = round(pair_matrix_a[i][j], 3)
45
46     for i in range(20):
47         sum = 0
48         for j in range(20):
49             sum += pair_matrix_a[i][j]
50         pair_matrix_a[i][20] = sum
51
52     # Sorting the matrix and amino acid in descending order of matrix score
    summed over each row
53     ans1 = sorted(zip([row[20] for row in pair_matrix_a],
54            [amino_acid_index[str(i)] for i in range(20)]), reverse=True)[:10]
55     ans1 = [item[1] for item in ans1]
56
57     #  (b) [Nij*100/(N-1)]
58     for i in range(20):
59         for j in range(20):
60             if pair_matrix_b[i][j] != 0:
61                 pair_matrix_b[i][j] *= 100
62                 pair_matrix_b[i][j] /= (len(seq) - 1)
63                 pair_matrix_b[i][j] = round(pair_matrix_b[i][j], 3)
64
65     for i in range(20):
66         sum = 0
67         for j in range(20):
68             sum += pair_matrix_b[i][j]
69         pair_matrix_b[i][20] = sum
```

```
70
71      ans2 = sorted(zip([row[20] for row in pair_matrix_b],
72              [amino_acid_index[str(i)] for i in range(20)]), reverse=True)[:10]
73      ans2 = [item[1] for item in ans2]
74
75      #  (c) [Nij*100/(Ni*Nj)]
76      for i in range(20):
77          for j in range(20):
78              if pair_matrix_c[i][j] != 0:
79                  pair_matrix_c[i][j] *= 100
80                  pair_matrix_c[i][j] /= (amino_freq[i] * amino_freq[j])
81                  pair_matrix_c[i][j] = round(pair_matrix_c[i][j], 3)
82
83      for i in range(20):
84          sum = 0
85          for j in range(20):
86              sum += pair_matrix_c[i][j]
87          pair_matrix_c[i][20] = sum
88
89      ans3 = sorted(zip([row[20] for row in pair_matrix_c],
90              [amino_acid_index[str(i)] for i in range(20)]), reverse=True)[:10]
91      ans3 = [item[1] for item in ans3]
92
93      return ans1, ans2, ans3
94
95  ans1_1, ans2_1, ans3_1 = pair_pref(seq1)
96  ans1_2, ans2_2, ans3_2 = pair_pref(seq2)
97  ans1_3, ans2_3, ans3_3 = pair_pref(seq3)
```

LISTING 5. Code to list top 10 preferred residues

Enclosed below are the tables that contain the top 10 residues obtained for each sequence using each of the given pair-preferences methods.

Top 10 residues in order from each of three pair preferences for Sequence 1			
Preference Number	Top residues in Method (a)	Top residues in Method (b)	Top residues in Method (c)
1	G	A	F
2	P	T	C
3	A	L	G
4	K	P	P
5	L	K	D
6	T	W	K
7	F	S	Y
8	C	R	V
9	S	G	N
10	R	Y	L

FIGURE 3. Top 10 preferred residues in Sequence 1

Top 10 residues in order from each of three pair preferences for Sequence 2			
Preference Number	Top residues in Method (a)	Top residues in Method (b)	Top residues in Method (c)
1	L	A	R
2	R	G	T
3	A	L	L
4	G	I	Q
5	I	V	P
6	V	M	K
7	M	F	V
8	P	P	G
9	Q	R	Y
10	F	Q	I

FIGURE 4. Top 10 preferred residues in Sequence 2

Top 10 residues in order from each of three pair preferences for Sequence 3			
Preference Number	Top residues in Method (a)	Top residues in Method (b)	Top residues in Method (c)
1	G	G	K
2	S	A	I
3	A	S	G
4	Y	D	Q
5	D	L	W
6	K	V	P
7	L	F	Y
8	I	Y	M
9	W	T	D
10	F	I	S

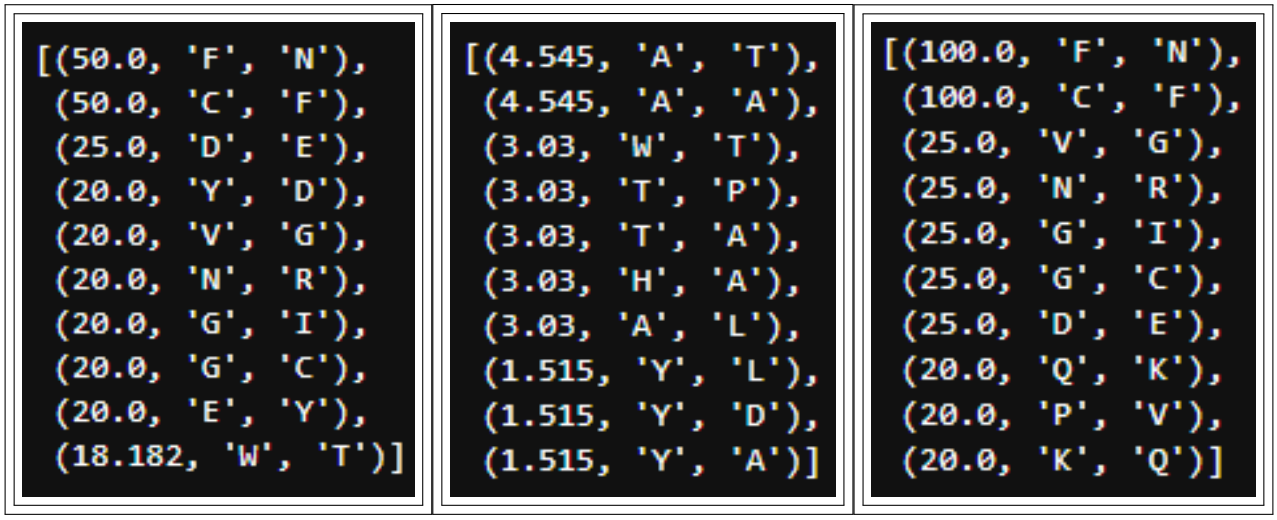
FIGURE 5. Top 10 preferred residues in Sequence 3

Above tables show the top 10 residues from each pair preference methods. The top 10 residue pairs in each of the given sequences can also be obtained. The code for the same is enclosed below along with the tabulated outputs.

```
# Obtaining the top 10 highest values in the given matrix for pair-preference maximization
def top_10_values(matrix):
    # Flatten the matrix and store the values along with their corresponding indices
```

```
    flattened = []
    for i in range(20):
        for j in range(20):
            flattened.append([matrix[i][j], amino_acid_index[str(i)],
            amino_acid_index[str(j)]])

    # Sort the flattened list based on the values in descending order
    sorted_values = sorted(zip([float(flattened[q][0]) for q in range
    (len(flattened))], [flattened[q][1] for q in range(len(flattened))],
    [flattened[q][2] for q in range(len(flattened))]), reverse=True)


    # Return the top 10 values along with their indices
    return sorted_values[:10]

aaa1 = top_10_values(aa1)
bbb1 = top_10_values(bb1)
ccc1 = top_10_values(cc1)

print(aaa1)
print(bbb1)
print(ccc1)
```

Listing 6. Code to compute top 10 residue pairs



(A) Seq1 (B) Seq2 (C) Seq3

Figure 6. Top 10 residue pairs in the 3 sequences with pair-preference scores

The 20x20 matrices for each sequence and each method are shown in the below images. The code used to generate each of these is a follow up from the above codes as follows:

```
from tabulate import tabulate

row_headings = amino_acid_sns
column_headings = amino_acid_sns
for i, row in enumerate(aaa):
    row.insert(0, row_headings[i])
aaa.insert(0, [''] + column_headings)
print(tabulate(aaa, headers='firstrow', tablefmt='simple'))
```

Listing 7. Code to print pair preference matrix

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	12.5	0	0	0	0	0	0	6.25	0	0	11.111	0	0	0	0	6.25	15.789	6.25	0	0
R	6.25	0	0	0	0	0	0	0	14.286	0	0	0	0	0	11.111	0	0	12.5	0	0
N	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	25	16.667	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	16.667	0	0	0	0	0	0	0	0
E	7.143	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0
G	0	0	0	0	20	0	0	0	0	20	10	0	0	0	0	0	0	0	14.286	0
H	13.333	0	0	0	0	0	0	0	16.667	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	16.667	0	0	0	0	0	0	0	0
L	5.556	0	0	12.5	0	0	0	0	11.111	0	8.333	9.091	0	0	0	0	7.692	0	0	0
K	5.882	0	0	0	0	16.667	0	0	0	0	0	0	0	0	10	11.111	0	11.111	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	14.286	0	0	0	0	0	0	0	0	11.111	8.333	11.111	0	16.667
S	0	12.5	0	0	0	0	0	0	0	0	10	11.111	0	0	0	0	0	0	14.286	0
T	10.526	9.091	0	0	0	0	0	9.091	0	0	0	8.333	0	0	16.667	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11.111	12.5	18.182	0	0	0
Y	6.667	0	0	20	0	0	0	0	0	0	11.111	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 7. Pair Preference matrix for Sequence 1, method a

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4.545	0	0	0	0	0	0	1.515	0	0	3.03	0	0	0	0	1.515	4.545	1.515	0	0
R	1.515	0	0	0	0	0	0	0	1.515	0	0	0	0	0	1.515	0	0	1.515	0	0
N	0	1.515	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	1.515	1.515	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	1.515	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	1.515	0	0	0	0	0	0	0	0
E	1.515	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.515	0
G	0	0	0	0	1.515	0	0	0	0	1.515	1.515	0	0	0	0	0	0	0	1.515	0
H	3.03	0	0	0	0	0	0	0	1.515	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	1.515	0	0	0	0	0	0	0	0
L	1.515	0	0	1.515	0	0	0	0	1.515	0	1.515	1.515	0	0	0	0	1.515	0	0	0
K	1.515	0	0	0	0	1.515	0	0	0	0	0	0	0	0	1.515	1.515	0	1.515	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	1.515	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	1.515	0	0	0	0	0	0	0	0	1.515	1.515	1.515	0	1.515
S	0	1.515	0	0	0	0	0	0	0	0	1.515	1.515	0	0	0	0	0	0	1.515	0
T	3.03	1.515	0	0	0	0	0	1.515	0	0	0	1.515	0	0	3.03	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.515	1.515	3.03	0	0	0
Y	1.515	0	0	1.515	0	0	0	0	0	0	1.515	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	1.515	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 8. Pair Preference matrix for Sequence 1, method b

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2.083	0	0	0	0	0	0	2.083	0	0	2.778	0	0	0	0	2.083	3.571	2.083	0	0
R	2.083	0	0	0	0	0	0	0	8.333	0	0	0	0	0	5	0	0	6.25	0	0
N	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	25	12.5	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
E	4.167	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16.667	0
G	0	0	0	0	25	0	0	0	0	25	4.167	0	0	0	0	0	0	0	8.333	0
H	5.556	0	0	0	0	0	0	0	11.111	0	0	0	0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
L	1.389	0	0	8.333	0	0	0	0	5.556	0	2.778	3.333	0	0	0	0	2.381	0	0	0
K	1.667	0	0	0	0	20	0	0	0	0	0	0	0	0	4	5	0	5	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	5	2.857	5	0	20
S	0	6.25	0	0	0	0	0	0	0	0	4.167	5	0	0	0	0	0	0	8.333	0
T	2.381	3.571	0	0	0	0	0	3.571	0	0	0	2.857	0	0	5.714	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	6.25	7.143	0	0	0
Y	2.778	0	0	16.667	0	0	0	0	0	0	5.556	0	0	0	0	0	0	0	0	0
V	0	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0

FIGURE 9. Pair Preference matrix for Sequence 1, method c

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	19.231	6.667	0	0	0	0	0	0	0	14.286	0	0	0	0	0	0	0	0	0	15.789
R	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	33.333	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	6.667	0	0	0	0	0	0	0	0	0	9.091	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	16.667	20	0	0	0	0	0
E	0	0	0	0	0	0	0	9.091	0	0	0	0	0	0	0	0	0	0	0	0
G	8.696	0	0	0	0	0	0	5	0	11.111	21.053	9.091	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	4.762	0	0	0	0	0	0	16.667	0	0	5.882	0	0	0	18.182	0	0	0	0	7.143
L	4.545	9.091	0	0	0	0	10	10.526	0	5.882	5.556	0	0	0	0	0	0	0	10	6.667
K	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0
M	0	0	0	0	0	0	0	13.333	0	7.692	0	0	10	11.111	0	0	0	0	0	0
F	5.882	0	0	0	0	0	0	0	0	8.333	7.692	0	0	12.5	0	0	0	0	0	0
P	0	0	0	20	0	0	0	0	0	0	8.333	0	12.5	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	33.333	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14.286
V	5.263	0	0	12.5	0	0	0	6.25	0	0	0	0	27.273	0	0	0	0	0	0	0

FIGURE 10. Pair Preference matrix for Sequence 2, method a

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	7.463	1.493	0	0	0	0	0	0	0	4.478	0	0	0	0	0	0	0	0	0	4.478
R	0	0	0	0	0	1.493	0	0	0	0	0	0	0	0	0	0	1.493	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	1.493	0	0	0	0	0	0	0	0	0	1.493	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	1.493	1.493	0	0	0	0	0
E	0	0	0	0	0	0	0	1.493	0	0	0	0	0	0	0	0	0	0	0	0
G	2.985	0	0	0	0	0	0	1.493	0	2.985	5.97	1.493	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	1.493	0	0	0	0	0	0	4.478	0	0	1.493	0	0	0	2.985	0	0	0	0	1.493
L	1.493	1.493	0	0	0	0	1.493	2.985	0	1.493	1.493	0	0	0	0	0	0	0	1.493	1.493
K	0	0	0	0	0	0	0	0	0	0	0	0	0	1.493	0	0	0	0	0	0
M	0	0	0	0	0	0	0	2.985	0	1.493	0	0	1.493	1.493	0	0	0	0	0	0
F	1.493	0	0	0	0	0	0	0	0	1.493	1.493	0	0	1.493	0	0	0	0	0	0
P	0	0	0	1.493	0	0	0	0	0	0	1.493	0	1.493	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	1.493	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.493
V	1.493	0	0	1.493	0	0	0	1.493	0	0	0	0	4.478	0	0	0	0	0	0	0

FIGURE 11. Pair Preference matrix for Sequence 2, method b

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2.959	3.846	0	0	0	0	0	0	0	2.885	0	0	0	0	0	0	0	0	0	3.846
R	0	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	50	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	3.846	0	0	0	0	0	0	0	0	0	5.556	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	12.5	16.667	0	0	0	0	0
E	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0
G	1.538	0	0	0	0	0	0	1	0	2.5	4.444	10	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
I	0.962	0	0	0	0	0	0	3.75	0	0	1.389	0	0	0	8.333	0	0	0	0	2.083
L	0.855	5.556	0	0	0	0	11.111	2.222	0	1.389	1.235	0	0	0	0	0	0	0	11.111	1.852
K	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	0	0	0
M	0	0	0	0	0	0	0	4	0	2.5	0	0	4	5	0	0	0	0	0	0
F	1.923	0	0	0	0	0	0	0	0	3.125	2.778	0	0	6.25	0	0	0	0	0	0
P	0	0	0	16.667	0	0	0	0	0	0	3.704	0	6.667	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
T	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16.667
V	1.282	0	0	8.333	0	0	0	1.667	0	0	0	0	10	0	0	0	0	0	0	0

FIGURE 12. Pair Preference matrix for Sequence 2, method c

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	9.375	0	4.762	4	0	0	0	15.385	0	4.167	0	0	0	4.167	5	0	4.348	0	0	4.167
R	4.762	0	0	0	0	0	0	0	0	7.692	14.286	0	0	0	0	0	0	0	0	7.692
N	4.762	0	0	7.143	0	0	0	3.571	0	0	7.143	0	0	7.692	0	0	0	0	0	0
D	4	7.143	0	0	0	7.143	6.667	3.125	0	0	0	6.667	0	0	0	4.167	0	6.667	0	5.882
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	10	0	0	0	0	0	0	0	0	0	0	0	11.111	10	8.333	0	0	0
E	4.545	0	0	6.667	0	0	0	6.897	0	0	0	0	0	7.143	0	4.762	0	0	0	0
G	7.692	3.571	0	9.375	0	7.143	0	2.174	0	6.452	9.375	3.448	4.167	0	0	2.632	3.333	0	6.452	6.452
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5.882	0	0	10	0
I	0	0	0	0	0	0	14.286	6.452	10	0	0	7.143	0	0	8.333	0	0	0	0	0
L	0	0	7.143	0	0	0	0	6.25	0	0	0	13.333	0	5.882	0	8.333	6.25	0	0	0
K	0	0	0	0	0	9.091	0	0	12.5	7.143	0	0	0	7.143	10	4.762	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12.5	0	0	0
F	12.5	0	0	0	0	0	7.143	0	0	6.25	0	0	0	0	0	4.348	0	0	6.25	6.25
P	0	0	11.111	0	0	0	0	0	0	8.333	0	0	0	8.333	0	0	0	0	0	8.333
S	3.226	0	5	0	0	0	0	5.263	0	4.348	12.5	0	0	4.348	0	3.333	4.545	9.524	4.348	4.348
T	0	8.333	0	0	0	0	0	10	0	0	0	0	0	0	0	4.545	0	0	13.333	0
W	0	0	0	20	0	9.091	8.333	0	0	0	0	0	0	0	0	0	0	8.333	0	0
Y	0	7.692	0	0	0	0	7.143	0	0	0	0	7.143	0	0	0	8.696	6.667	14.286	0	0
V	4.167	7.692	0	0	0	0	0	9.677	0	0	0	0	0	6.25	0	4.348	0	0	6.25	0

FIGURE 13. Pair Preference matrix for Sequence 3, method a

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	2	0	0.667	0.667	0	0	0	4	0	0.667	0	0	0	0.667	0.667	0	0.667	0	0	0.667
R	0.667	0	0	0	0	0	0	0	0	0.667	1.333	0	0	0	0	0	0	0	0	0.667
N	0.667	0	0	0.667	0	0	0	0.667	0	0	0.667	0	0	0.667	0	0	0	0	0	0
D	0.667	0.667	0	0	0	0.667	0.667	0.667	0	0	0	0.667	0	0	0	0.667	0	0.667	0	0.667
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0.667	0	0	0	0	0	0	0	0	0	0	0	0.667	1.333	0.667	0	0	0
E	0.667	0	0	0.667	0	0	0	1.333	0	0	0	0	0	0.667	0	0.667	0	0	0	0
G	2	0.667	0	2	0	1.333	0	0.667	0	1.333	2	0.667	0.667	0	0	0.667	0.667	0	1.333	1.333
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.667	0	0	0.667	0
I	0	0	0	0	0	0	1.333	1.333	0.667	0	0	0.667	0	0	0.667	0	0	0	0	0
L	0	0	0.667	0	0	0	0	1.333	0	0	0	1.333	0	0.667	0	1.333	0.667	0	0	0
K	0	0	0	0	0	0.667	0	0	0.667	0.667	0	0	0	0.667	0.667	0.667	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.667	0	0	0
F	2	0	0	0	0	0	0.667	0	0	0.667	0	0	0	0	0	0.667	0	0	0.667	0.667
P	0	0	0.667	0	0	0	0	0	0	0.667	0	0	0	0.667	0	0	0	0	0	0.667
S	0.667	0	0.667	0	0	0	0	1.333	0	0.667	2	0	0	0.667	0	0.667	0.667	1.333	0.667	0.667
T	0	0.667	0	0	0	0	0	2	0	0	0	0	0	0	0	0.667	0	0	1.333	0
W	0	0	0	2	0	0.667	0.667	0	0	0	0	0	0	0	0	0	0	0.667	0	0
Y	0	0.667	0	0	0	0	0.667	0	0	0	0	0.667	0	0	0	1.333	0.667	1.333	0	0
V	0.667	0.667	0	0	0	0	0	2	0	0	0	0	0	0.667	0	0.667	0	0	0.667	0

FIGURE 14. Pair Preference matrix for Sequence 3, method b

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	1.172	0	1.25	0.694	0	0	0	1.63	0	0.781	0	0	0	0.781	1.562	0	0.893	0	0	0.781
R	1.25	0	0	0	0	0	0	0	0	2.5	4.444	0	0	0	0	0	0	0	0	2.5
N	1.25	0	0	2.222	0	0	0	0.87	0	0	2.222	0	0	2.5	0	0	0	0	0	0
D	0.694	2.222	0	0	0	2.222	1.852	0.483	0	0	0	1.852	0	0	0	0.741	0	1.852	0	1.389
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	4	0	0	0	0	0	0	0	0	0	0	0	5	2.667	2.857	0	0	0
E	1.042	0	0	1.852	0	0	0	1.449	0	0	0	0	0	2.083	0	1.111	0	0	0	0
G	0.815	0.87	0	1.449	0	1.739	0	0.189	0	1.087	1.449	0.725	4.348	0	0	0.29	0.621	0	1.087	1.087
H	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.333	0	0	6.25	0
I	0	0	0	0	0	0	4.167	1.087	6.25	0	0	2.083	0	0	3.125	0	0	0	0	0
L	0	0	2.222	0	0	0	0	0.966	0	0	0	3.704	0	1.389	0	1.481	1.587	0	0	0
K	0	0	0	0	0	3.333	0	0	8.333	2.083	0	0	0	2.083	4.167	1.111	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	14.286	0	0	0
F	2.344	0	0	0	0	0	2.083	0	0	1.562	0	0	0	0	0	0.833	0	0	1.562	1.562
P	0	0	5	0	0	0	0	0	0	3.125	0	0	0	3.125	0	0	0	0	0	3.125
S	0.417	0	1.333	0	0	0	0	0.58	0	0.833	2.222	0	0	0.833	0	0.444	0.952	2.222	0.833	0.833
T	0	2.857	0	0	0	0	0	1.863	0	0	0	0	0	0	0	0.952	0	0	3.571	0
W	0	0	0	5.556	0	3.333	2.778	0	0	0	0	0	0	0	0	0	0	2.778	0	0
Y	0	2.5	0	0	0	0	2.083	0	0	0	0	2.083	0	0	0	1.667	1.786	4.167	0	0
V	0.781	2.5	0	0	0	0	0	1.63	0	0	0	0	0	1.562	0	0.833	0	0	1.562	0

FIGURE 15. Pair Preference matrix for Sequence 3, method c

Question 5. Compute average hydrophobicity (Hgm), Helical contact area (Ca) and Total non-bonded energy (Et) for the sequences in Q1 and comment on the results. (Refer www.iitm.ac.in/bioinfo/fold_rate/prop_orig.html for the properties).

Solution. The given website displays a lot of properties with respect to amino acids. So firstly, I took the rows with the above three properties of **Hgm**, **Ca**, and **Et** and created a matrix with those values stored in each row. Now, this matrix will be used to calculate the average scores for the sequences given in question 1.
The matrices for the above given three properties for all the amino acids can be tabulated and plotted as follows:

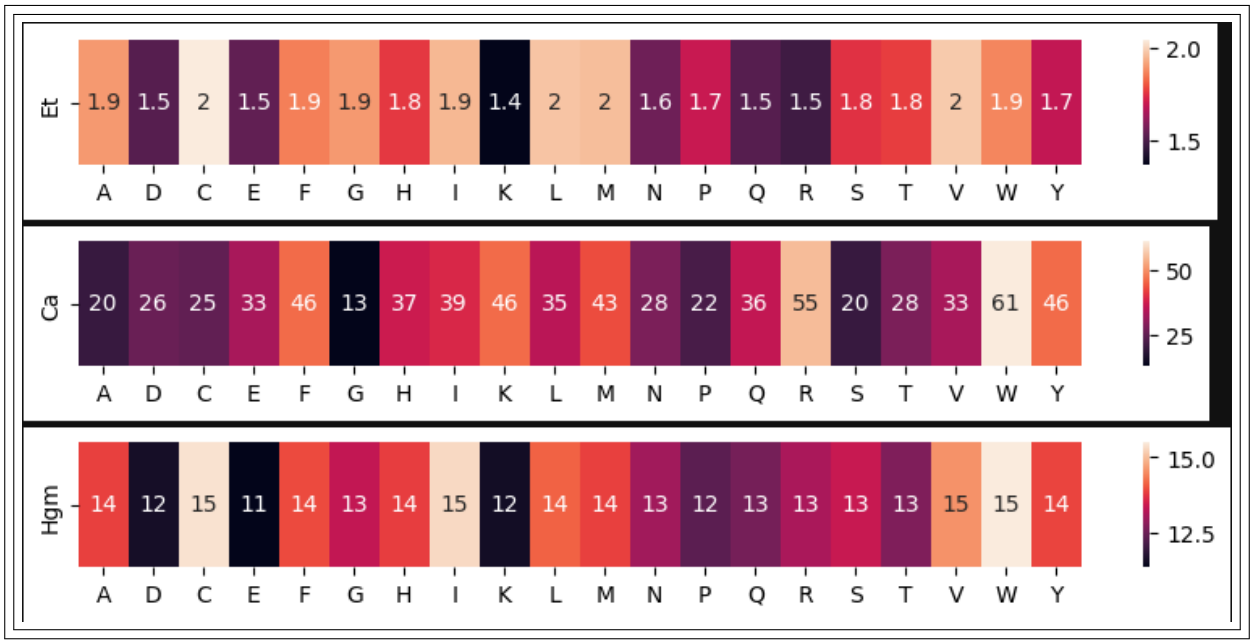


FIGURE 16. Matrices for given 3 properties plotted

The code to store the tabulated data as stored above is given below. Also enclosed is the code to compute the average of these properties mentioned in the question.

```
# Order of amino acids in website
property_aa =
{"A":  0, "D":  1, "C":  2, "E":  3, "F":  4,
 "G":  5, "H":  6, "I":  7, "K":  8, "L":  9,
 "M": 10, "N": 11, "P": 12, "Q": 13, "R": 14,
 "S": 15, "T": 16, "V": 17, "W": 18, "Y": 19}

property = {"Et": 0, "Ca": 1, "Hgm": 2}

# Reading the property matrix from the text file
property_matrix = []
with open("property.txt", 'r') as f:
    line = f.read()
    line_list = line.split("\n")

    for i in range(len(line_list)):
        line_i = line_list[i].split(" ")
        line_i = [float(value) for value in line_i if value != ""]
        property_matrix.append(line_i)

# Code to compute the average scores for the three properties
```

```
def compute_avg_scores(seq):
    hgm_score = 0
    ca_score = 0
    et_score = 0

    # iterate through the length of sequence and add appropriately
    for i in range(len(seq)):
        hgm_score += property_matrix[2][property_aa[seq[i]]]
        ca_score  += property_matrix[1][property_aa[seq[i]]]
        et_score  += property_matrix[0][property_aa[seq[i]]]

    # Average the scores
    hgm_score /= len(seq)

    return hgm_score, ca_score, et_score

hgm1, ca1, et1 = compute_avg_scores(seq1)
hgm2, ca2, et2 = compute_avg_scores(seq2)
hgm3, ca3, et3 = compute_avg_scores(seq3)

print(hgm1, ca1, et1)
print(hgm2, ca2, et2)
print(hgm3, ca3, et3)
```

LISTING 8. Code to compute the average of properties

The output generated from the above code for the three sequences is as follows:

```
13.352537313432844 2156.0 117.74000000000005
13.77161764705882 2067.0 126.66000000000003
13.418675496688737 4616.0 267.7500000000001
```

FIGURE 17. Output average properties

Properties for the Sequence 1		
Average Hgm value	Total Ca value	Total Et value
13.352537313432844	2156.0	117.74000000000005

Properties for the Sequence 2		
Average Hgm value	Total Ca value	Total Et value
13.77161764705882	2067.0	126.66000000000003

Properties for the Sequence 3		
Average Hgm value	Total Ca value	Total Et value
13.418675496688737	4616.0	267.7500000000001

The comments on the results are as follows:

- Sequence 2 has the highest average hydrophobicity value (though very marginally), indicating a potentially higher tendency to interact with hydrophobic regions.
- Sequence 3 has the highest helical contact area, suggesting a larger interface for helical interactions, which is evident from the fact that Seq 3 is the longest.
- Owing to the large size of sequence3 in comparison to others, it has the highest total non-bonded energy, indicating a potential of having great interaction with surrounding molecules.