Practical11

Name: Kudari Pavani
Roll no: BE19B023

1. Algorithm:

- The sequence and secondary structure identification are given to the programme as two strings.
- Determine the number of times each AA appears in the sequence.
- Count the number of times each AA appears in the sequence for those whose secondary structural identity is Helix (H).
- For each AA, calculate the ratio of the two values above.
- Calculate the ratio of the total number of AAs in helix conformation to the input sequence length.
- Each AA's tendency is the ratio of the two ratios above.

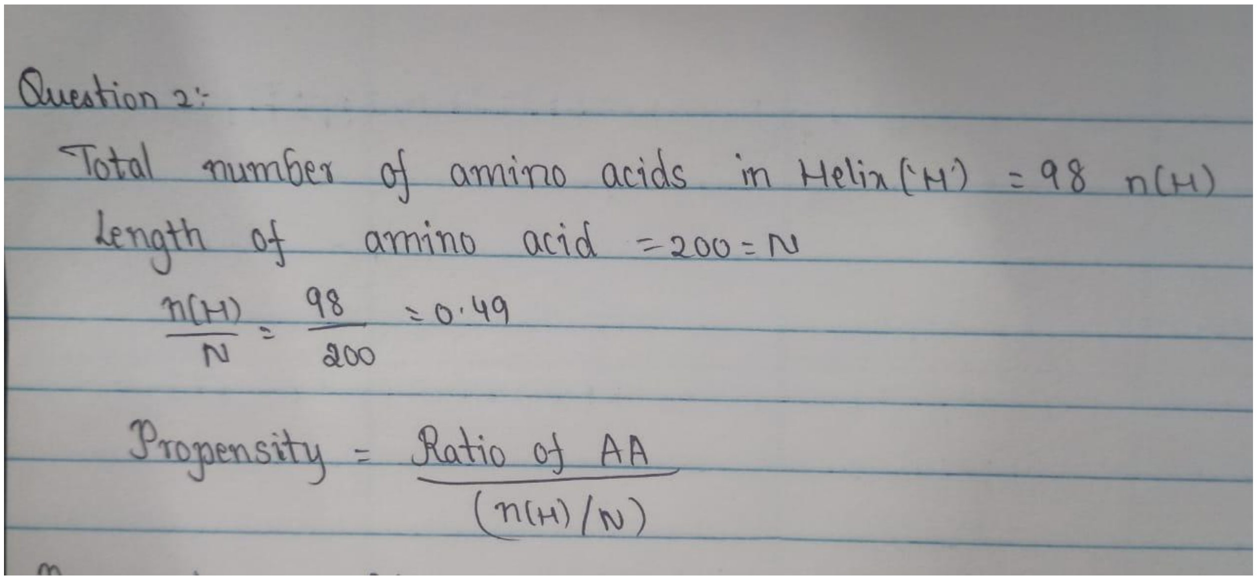
```
def Q1(Seq, SS):
    AA = ['A', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'K', 'L','M', 'N', 'P', 'Q', 'R', 'S', 'T', 'V', 'W', 'Y']
    Composition,AA_H = [0]*20, [0]*20
    p = [0]*20

    for i in range(len(Seq)):
        Composition[AA.index(Seq[i])] += 1
        if SS[i] == 'H':
            AA_H[AA.index(Seq[i])] += 1

    print('Propensity:\n')
    for i in range(20):
        if Composition[i] != 0:
            p[i] = (AA_H[i]/Composition[i])/(B.count('H')/len(Seq))
            print(f"{AA[i]}\t:\t{p[i]}")


Seq =
'LGASGIAAFAFGSTAILIILFNMAAEVHFDPLQFFRQFFWLGLYPPKAQYGMGIPPLHDGGWWL
MAGLFMTLSLGSWWIRVYSRARALGLGTHIAWNFAAAIFFVLCIGCIHPTLVGSWSEGVPFGIW
PHIDWLTAFSIRYGNFYYCPWHGFSIGFAYGCGLLFAAHGATILAVARFGGDREIEQITDRGTAVE
RAALFW'
SS =
'XHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXH
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHXXHHHHHHHHHHHHHHHHHXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXX
XXXXXXXXXXXXXXXXXX'
Q1(Seq, SS)
```

Output:

Propensity:

A	:	1.5510204081632653
C	:	1.0204081632653061
D	:	0.0
E	:	0.40816326530612246
F	:	1.0204081632653061
G	:	1.0612244897959184
H	:	0.8746355685131195
I	:	1.2004801920768309
K	:	0.0
L	:	1.2244897959183674
M	:	1.530612244897959
N	:	1.3605442176870748
P	:	0.22675736961451246
Q	:	0.0
R	:	0.6802721088435374
S	:	1.3605442176870748
T	:	0.7653061224489796
V	:	0.5830903790087464
W	:	1.1131725417439702
Y	:	0.5830903790087464

Question 2:



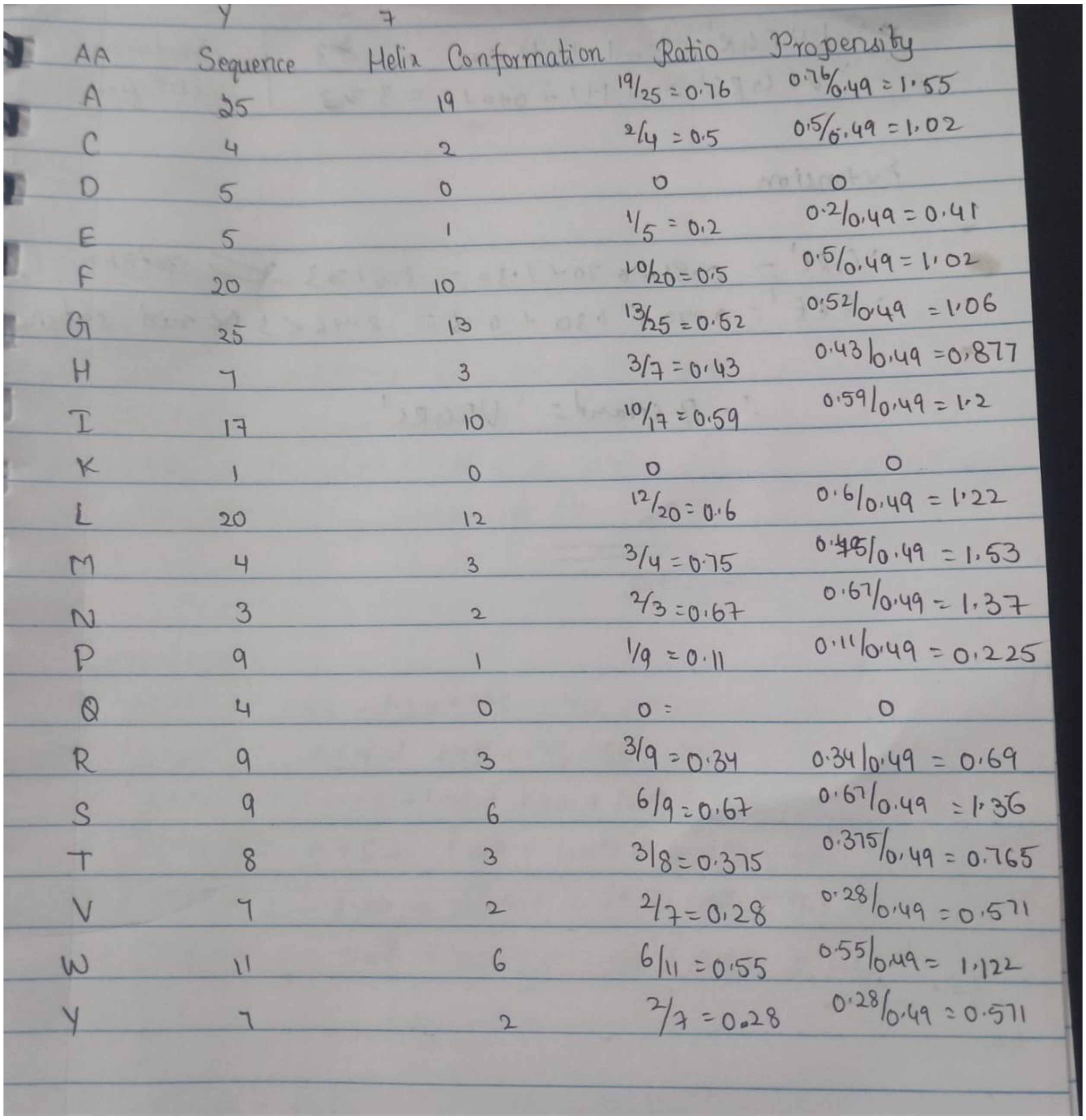
Question 2:-

Total number of amino acids in Helix('H') = 98 n(H)

Length of amino acid = 200 = N

$$\frac{n(H)}{N} = \frac{98}{200} = 0.49$$

$$\text{Propensity} = \frac{\text{Ratio of AA}}{(n(H)/N)}$$



AA	Sequence	Helix Conformation	Ratio	Propensity
A	25	19	19/25 = 0.76	0.76/0.49 = 1.55
C	4	2	2/4 = 0.5	0.5/0.49 = 1.02
D	5	0	0	0
E	5	1	1/5 = 0.2	0.2/0.49 = 0.41
F	20	10	10/20 = 0.5	0.5/0.49 = 1.02
G	25	13	13/25 = 0.52	0.52/0.49 = 1.06
H	7	3	3/7 = 0.43	0.43/0.49 = 0.877
I	17	10	10/17 = 0.59	0.59/0.49 = 1.2
K	1	0	0	0
L	20	12	12/20 = 0.6	0.6/0.49 = 1.22
M	4	3	3/4 = 0.75	0.75/0.49 = 1.53
N	3	2	2/3 = 0.67	0.67/0.49 = 1.37
P	9	1	1/9 = 0.11	0.11/0.49 = 0.225
Q	4	0	0 =	0
R	9	3	3/9 = 0.34	0.34/0.49 = 0.69
S	9	6	6/9 = 0.67	0.67/0.49 = 1.36
T	8	3	3/8 = 0.375	0.375/0.49 = 0.765
V	7	2	2/7 = 0.28	0.28/0.49 = 0.571
W	11	6	6/11 = 0.55	0.55/0.49 = 1.122
Y	7	2	2/7 = 0.28	0.28/0.49 = 0.571

Question 3:

```
def Q2_matching(a, b):
    match = ''
    for i in range(len(a)):
        for j in range(len(b)):
            d = 1
            while i+d <= len(a) and j+d <= len(b) and a[i:i+d] == b[j:j+d]:
                if len(match) <= len(a[i:i+d]):
                    match = a[i:i+d]
            d += 1
    return match

def Q2(Seq):
    helix = {'A': 'Ha', 'C': 'ia', 'D': 'ia', 'E': 'Ha', 'F': 'ha',
             'G': 'Ba', 'H': 'ha', 'I': 'Ia', 'K': 'Ia', 'L': 'Ha',
             'M': 'ha', 'N': 'ba', 'P': 'Ba', 'Q': 'ha', 'R': 'ia',
             'S': 'ia', 'T': 'ia', 'V': 'ha', 'W': 'ha', 'Y': 'ba'}

    strand = {'A': 'Ib', 'C': 'hb', 'D': 'ib', 'E': 'Bb', 'F': 'hb',
              'G': 'ib', 'H': 'bb', 'I': 'Hb', 'K': 'bb', 'L': 'hb',
              'M': 'Hb', 'N': 'bb', 'P': 'bb', 'Q': 'hb', 'R': 'ib',
              'S': 'bb', 'T': 'hb', 'V': 'Hb', 'W': 'hb', 'Y': 'hb'}

    ph = {'A': 1.45, 'C': 0.77, 'D': 0.98, 'E': 1.53, 'F': 1.12,
          'G': 0.53, 'H': 1.24, 'I': 1.00, 'K': 1.07, 'L': 1.34,
          'M': 1.20, 'N': 0.73, 'P': 0.59, 'Q': 1.17, 'R': 0.79,
          'S': 0.79, 'T': 0.82, 'V': 1.14, 'W': 1.14, 'Y': 0.61}

    pb = {'A': 0.97, 'C': 1.30, 'D': 0.80, 'E': 0.26, 'F': 1.28,
          'G': 0.81, 'H': 0.71, 'I': 1.60, 'K': 0.74, 'L': 1.22,
          'M': 1.67, 'N': 0.65, 'P': 0.62, 'Q': 1.23, 'R': 0.90,
          'S': 0.72, 'T': 1.20, 'V': 1.65, 'W': 1.19, 'Y': 1.29}

    cf = {'Ha': 1, 'ha': 1, 'Ia': 0.5, 'ia': 0, 'ba': -1, 'Ba': -1,
          'Hb': 1, 'hb': 1, 'Ib': 0.5, 'ib': 0, 'bb': -1, 'Bb': -1}

    helix_seq = []
    strand_seq = []

    print('\nAlpha Helices:')
    i = 0
    while i < len(A) - 6:
        value = 0
        for j in range(6):
            value += cf[helix[A[i:i+6][j]]]
        if value >= 4:
            done = 1
            k = 0
            while done == 1:
                next_seg = A[i+k+2:i+k+6]
                p = 0
                for l in range(4):
                    p += ph[next_seg[l]]
                if p < 4.00:
                    done = 0
                else:
                    k += 1
            if k == 0:
                print(A[i:i + k + 6])
                helix_seq.append(A[i:i + k + 6])
                i = i + k + 6
            else:
                print(A[i:i + k + 5])
                helix_seq.append(A[i:i + k + 5])
                i = i + k + 5
        else:
            i += 1

    print('\nBeta Strands')
    i1 = 0
    while i1 < len(Seq) - 5:
        value = 0
        for j in range(5):
            value += cf[strand[A[i1:i1+5][j]]]
        if value >= 3:
            done = 1
            k = 0
            while done == 1 and (i1+k+5) <= len(A):
                next_seg = A[i1+k+2:i1+k+5]
                prop = 0
                for l in range(3):
                    prop += pb[next_seg[l]]
                if prop < 3.00:
                    done = 0
                else:
                    k += 1
            if k == 0:
                print(A[i1:i1 + k + 5])
                strand_seq.append(A[i1:i1 + k + 5])
                i1 = i1 + k + 5
            else:
                print(A[i1:i1 + k + 4])
                strand_seq.append(A[i1:i1 + k + 4])
                i1 = i1 + k + 4
        else:
            i1 += 1
```

```
hf = []
sf = []
print('\nCommon segments ')
for i in range(len(helix_seq)):
    h = helix_seq[i]
    for j in range(len(strand_seq)):
        s = strand_seq[j]
        c = Q2_matching(h, s)
        match = len(c)
        if match != 0 and match >= 5:
            print('Helix - %s, Strand - %s, Common segment - %s' % (h, s, c))
            prop_helix = 0
            prop_strand = 0
            for k in range(match):
                prop_helix += ph[c[k]]
                prop_strand += pb[c[k]]
            if prop_helix > prop_strand:
                hf.append([j, h])
            else:
                sf.append([i, s])
for i in range(len(hf) - 1, -1, -1):
    a = strand_seq[hf[i][0]]
    strand_seq.remove(a)
for j in range(len(sf) - 1, -1, -1):
    b = helix_seq[sf[j][0]]
    helix_seq.remove(b)


print('\nFinal list of secondary structure segments after comparing -')
print('\nAlpha Helix segments')
for i in range(len(helix_seq)):
    print(helix_seq[i])
print('\nBeta Strand segments')
for i in range(len(strand_seq)):
    print(strand_seq[i])
```

Seq =
"KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSR
WWCNDGRTPGSRNLCNIPCSALLSSDITASVNC"
Q2(A)

OUTPUT:

Alpha Helices:
RCELAAAMKRH
WVCAAKFESNF

Beta Strands
VFGRC
LAAAMKR
WVCAA
TDYGILQIN

Common segments - Conflicting sequence
Helix - RCELAAAMKRH, Strand - LAAAMKR, Common segment - LAAAMKR
Helix - WVCAAKFESNF, Strand - WVCAA, Common segment - WVCAA

Final list of secondary structure segments after comparing -

Alpha Helix segments
RCELAAAMKRH

Beta Strand segments
VFGRC
WVCAA
TDYGILQIN

In [ ]:

Question 4:-

Alpha Helix:

KVFGRC :→ $0.5+1+1-1+0+0 = 1.5 < 4$

VFGRCE → $1+1-1+0+0+1 = 2 < 4$

FGRCEL → $1-1+0+0+1+1 = 2 < 4$

GRCELA → $-1+0+0+1+1+1 = 2 < 4$

RCELAA → $0+0+1+1+1+1 = 4 \geq 4$ (satisfied)

Extending the segment:

ELAA: $1.53+1.34+1.45+1.45 = 5.77 \geq 4$

LAAN: $1.34+1.45+1.45+1.20 = 5.69 \geq 4$

AAMK: $1.45+1.45+1.20+1.07 = 5.55 \geq 4$

AMKR: $1.45+1.20+1.07+0.79 = 4.51 \geq 4$

MKRH: $1.20+1.07+0.79+1.24 = 4.3 \geq 4$

KRHG: $1.07+0.79+1.24+0.53 = 3.63 \leq 4$

Hence the potential alpha helix segment = RCELAAMKRH

Beta strand:

KVFGR $= -1+1+1+0+0 = 1 < 3$

VFGRC $= 1+1+0+0+1 = 3 \geq 3$ (satisfied)

Extending the segment:

GRC $= 0.81+0.90+1.30 = 3.01 \geq 3$

RCE $= 0.90+1.30+0.26 = 2.46 \leq 3$

Hence the potential beta strand = VFGRC