Assignment 8

1. For obtaining the hydrophobicity profile, plot the values plot the position of the amino acid in X axis and the hydrophobicity of that amino acid in Y axis and plot the graph.
   For obtaining the alpha and beta sheets, first we need to make the mean of the hydrophobicity profile values and then make a binary array with zeroes and ones for the ones which has the hydrophobicity values that are lower and higher than the mean value.
   For alpha helices, we need to match them with either [0 0 1 1] or else [1 1 0 0] to determine the position of the alpha helix.
   For beta sheets, we match the sequence with the alternating 0's and 1's and then determine the ones which have equal to or more than 4 matches (ex: 0101… or 1010…).

   The python program for the given question is given as follows:

```
import matplotlib.pyplot as plt
import numpy as np


def hydrophobicityplot(string):
    hydrophobicity_values={'A': 13.85, 'D': 11.61, 'C': 15.37,
'E': 11.38, 'F': 13.93, 'G': 13.34,
                           'H': 13.82, 'I': 15.28, 'K': 11.58,
'L': 14.13, 'M': 13.86, 'N': 13.02,
                           'P': 12.35, 'Q': 12.61, 'R': 13.10,
'S': 13.39, 'T': 12.70, 'V': 14.56,
                           'W': 15.48, 'Y': 13.88}
    X=[]
    Y=[]
    for i in range(len(string)):
        X.append(i)
        Y.append(hydrophobicity_values[string[i]])
    plt.plot(X, Y)
    Ymean=sum(Y)/len(Y)
    Ymeanarr=[Ymean for i in range(len(Y))]
    plt.plot(X, Ymeanarr, c='r')
    plt.show()
    UpDownArr=[]
    for i in range(len(Y)):
        if Y[i]>Ymean:
            UpDownArr.append(1)
        else:
            UpDownArr.append(0)
    print(UpDownArr)
    #Alpha Helix
    print('Alpha Strands:')
    for i in range(len(Y)-3):
        arr1=[1, 1, 0, 0]
        arr2=[0, 0, 1, 1]
```

```
arr=[UpDownArr[i],UpDownArr[i+1],UpDownArr[i+2],UpDownArr[i+3]
]
        if (arr==arr1 or arr==arr2):
            print('Position {} to {} -'.format(i+1, i+4),
np.array(arr))
    #Beta Barrel
    print('Beta barrel')
    for i in range(len(Y)-4):
        arr1=[1, 0, 1, 0]
        arr2=[0, 1, 0, 1]

arr=[UpDownArr[i],UpDownArr[i+1],UpDownArr[i+2],UpDownArr[i+3]
]
        if (arr==arr1 or arr==arr2):
            for j in range(i+4,len(Y)):
                if j%2==0:
                    if arr[0:4]==arr1 and UpDownArr[j]==1:
                        arr.append(UpDownArr[j])
                    else:
                        break
                    if arr[0:4]==arr2 and UpDownArr[j]==0:
                        arr.append(UpDownArr[j])
                    else:
                        break
                else:
                    if arr[0:4]==arr1 and UpDownArr[j]==0:
                        arr.append(UpDownArr[j])
                    else:
                        break
                    if arr[0:4]==arr2 and UpDownArr[j]==1:
                        arr.append(UpDownArr[j])
                    else:
                        break
            print('Position {} to {}'.format(i+1,
len(arr)+i+1), np.array(arr))

if __name__=="__main__":
    file=open('Q1.fasta')
    #file=open('input.txt')
    data=file.read().splitlines()
    file.close()

a=''
strings=[]
head=[]
for i in data:
    if i[0]=='>':
        head.append(i)
        strings.append(a)
        a=''
```

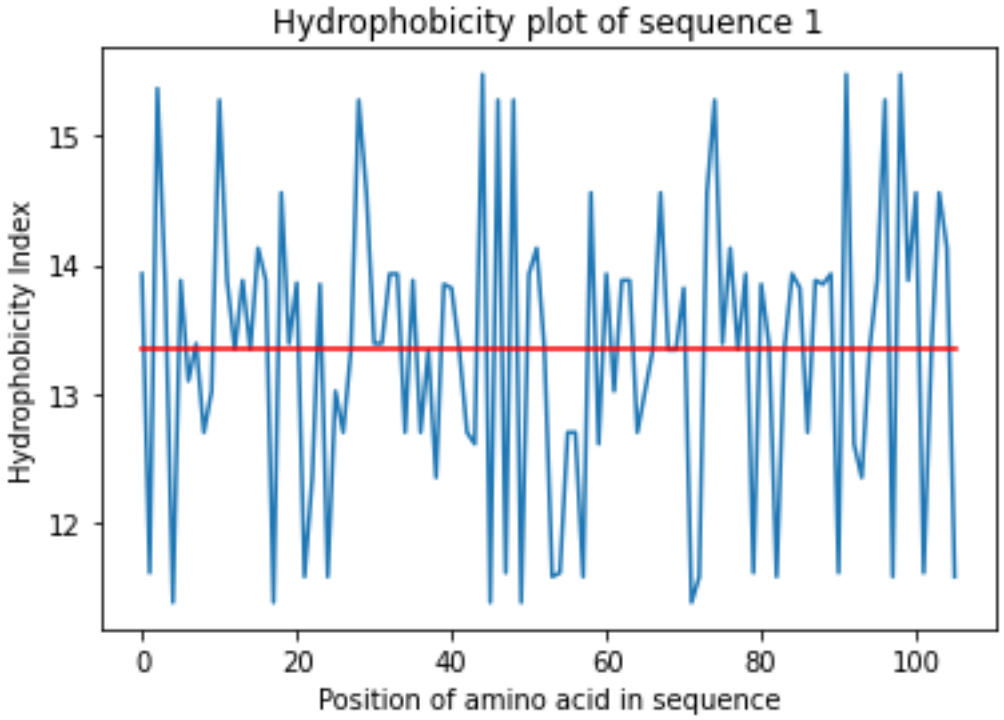
```
    else:
        a+=str(i)
strings.append(a)
strings.pop(0)
for i in range(len(strings)):
    strings[i]=strings[i].replace('\n', '')
for i in range(len(head)):
    head[i]=head[i].replace('\n','')

for i in range(len(strings)):
    hydrophobicityplot(strings[i])
```

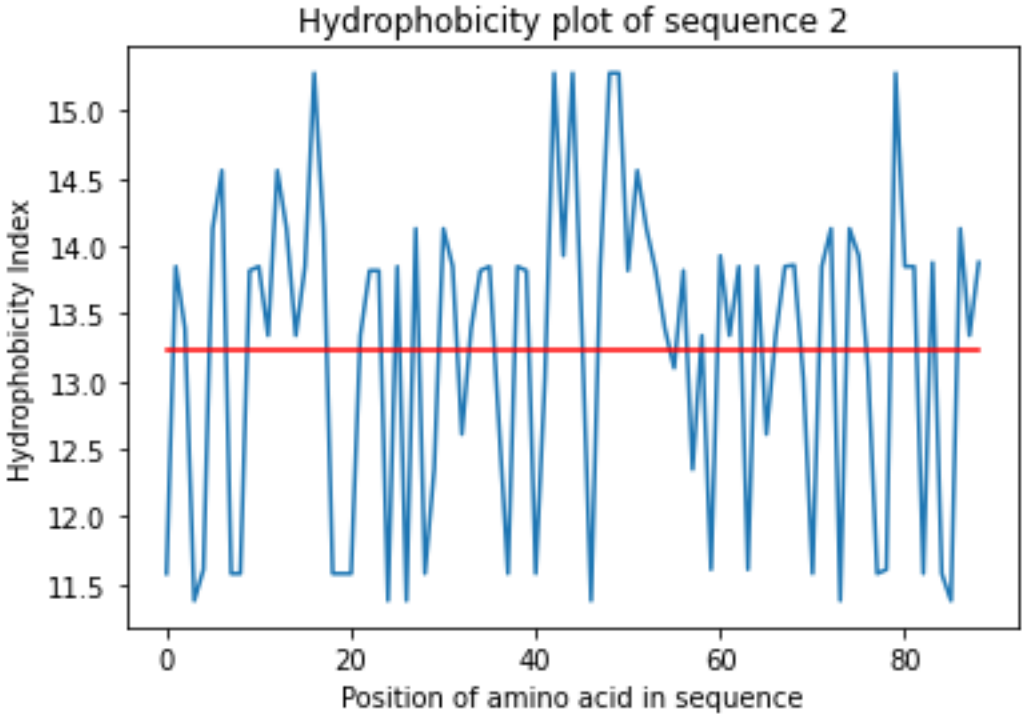
Output:

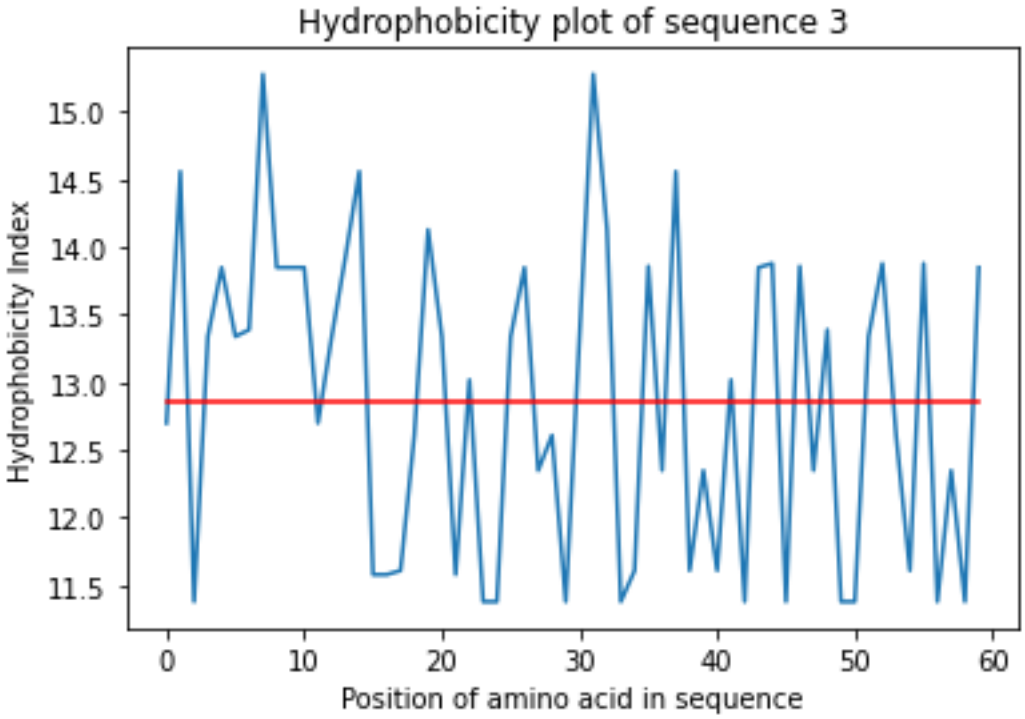
```
Alpha Strands:
Position 9 to 12 - [0 0 1 1]
Position 20 to 23 - [1 1 0 0]
Position 27 to 30 - [0 0 1 1]
Position 38 to 41 - [0 0 1 1]
Position 40 to 43 - [1 1 0 0]
Position 51 to 54 - [1 1 0 0]
Position 63 to 66 - [1 1 0 0]
Position 72 to 75 - [0 0 1 1]
Position 81 to 84 - [1 1 0 0]
Position 83 to 86 - [0 0 1 1]
Position 94 to 97 - [0 0 1 1]
Position 100 to 103 - [1 1 0 0]
Position 102 to 105 - [0 0 1 1]
Beta barrel
Position 4 to 8 [1 0 1 0]
Position 5 to 9 [0 1 0 1]
Position 6 to 11 [1 0 1 0 0]
Position 12 to 16 [1 0 1 0]
Position 13 to 17 [0 1 0 1]
Position 34 to 39 [1 0 1 0 0]
Position 44 to 48 [0 1 0 1]
Position 45 to 50 [1 0 1 0 1]
Position 46 to 50 [0 1 0 1]
Position 47 to 52 [1 0 1 0 1]
Position 48 to 52 [0 1 0 1]
Position 58 to 62 [0 1 0 1]
Position 59 to 64 [1 0 1 0 1]
Position 60 to 64 [0 1 0 1]
Position 77 to 82 [1 0 1 0 1]
Position 78 to 82 [0 1 0 1]
Position 90 to 95 [1 0 1 0 0]
[0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0,
0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0,
1, 0, 0, 1, 1, 1]
```

```
Alpha Strands:
Position 2 to 5 - [1 1 0 0]
Position 4 to 7 - [0 0 1 1]
Position 6 to 9 - [1 1 0 0]
Position 8 to 11 - [0 0 1 1]
Position 17 to 20 - [1 1 0 0]
Position 20 to 23 - [0 0 1 1]
Position 29 to 32 - [0 0 1 1]
Position 35 to 38 - [1 1 0 0]
Position 37 to 40 - [0 0 1 1]
Position 39 to 42 - [1 1 0 0]
Position 41 to 44 - [0 0 1 1]
Position 68 to 71 - [1 1 0 0]
Position 70 to 73 - [0 0 1 1]
Position 75 to 78 - [1 1 0 0]
Position 78 to 81 - [0 0 1 1]
Position 85 to 88 - [0 0 1 1]
Beta barrel
Position 24 to 28 [1 0 1 0]
Position 25 to 29 [0 1 0 1]
Position 26 to 31 [1 0 1 0 0]
Position 55 to 60 [1 0 1 0 1]
Position 56 to 60 [0 1 0 1]
Position 57 to 62 [1 0 1 0 1]
Position 58 to 62 [0 1 0 1]
Position 63 to 68 [1 0 1 0 1]
Position 64 to 68 [0 1 0 1]
Position 82 to 87 [1 0 1 0 0]
[0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1,
1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,
1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1]
Alpha Strands:
Position 14 to 17 - [1 1 0 0]
Position 18 to 21 - [0 0 1 1]
Position 24 to 27 - [0 0 1 1]
Position 26 to 29 - [1 1 0 0]
Position 29 to 32 - [0 0 1 1]
Position 32 to 35 - [1 1 0 0]
Position 50 to 53 - [0 0 1 1]
Position 52 to 55 - [1 1 0 0]
Beta barrel
Position 1 to 5 [0 1 0 1]
Position 21 to 25 [1 0 1 0]
Position 35 to 39 [0 1 0 1]
Position 36 to 41 [1 0 1 0 1]
Position 41 to 45 [0 1 0 1]
Position 45 to 50 [1 0 1 0 1]
Position 46 to 50 [0 1 0 1]
Position 47 to 51 [1 0 1 0]
```

Plots:


Hydrophobicity plot of sequence 1
Hydrophobicity Index
15
14
13
12
0
20
40
60
80
100
Position of amino acid in sequence


Hydrophobicity plot of sequence 2
Hydrophobicity Index
15.0
14.5
14.0
13.5
13.0
12.5
12.0
11.5
0
20
40
60
80
Position of amino acid in sequence



2. Amphipathic index is calculated in similar way as hydrophobicity plot but the difference comes in the alpha and beta parts and calculation of the amphipathic index. In this, the alpha helix and beta strand length is already given in the query and we calculate the amphipathic index by using the absolute difference between the higher and lower values of the corresponding beta sheets hydrophobicity values…

   The python program for the given question is given as follows:

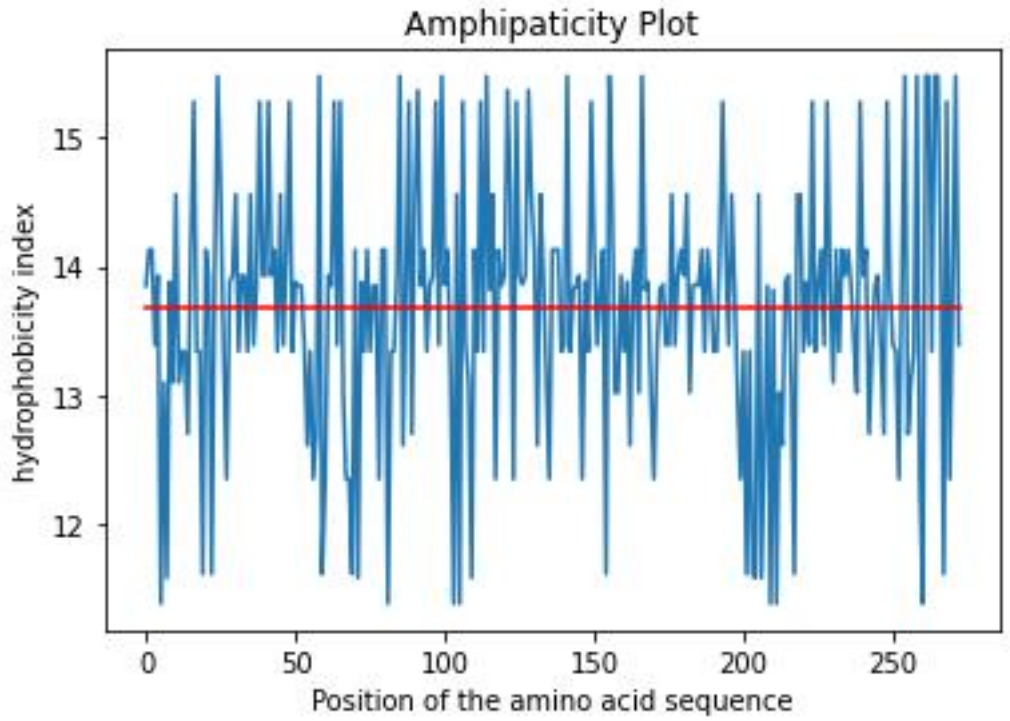
```
import matplotlib.pyplot as plt
import numpy as np


def amphipaticityplot(string):
    hydrophobicity_values={'A': 13.85, 'D': 11.61, 'C': 15.37,
'E': 11.38, 'F': 13.93, 'G': 13.34,
                           'H': 13.82, 'I': 15.28, 'K': 11.58,
'L': 14.13, 'M': 13.86, 'N': 13.02,
                           'P': 12.35, 'Q': 12.61, 'R': 13.10,
'S': 13.39, 'T': 12.70, 'V': 14.56,
                           'W': 15.48, 'Y': 13.88}
    X=[]
    Y=[]
    for i in range(len(string)):
        X.append(i)
        Y.append(hydrophobicity_values[string[i]])
    plt.plot(X, Y)
    Ymean=sum(Y)/len(Y)
    Ymeanarr=[Ymean for i in range(len(Y))]
    plt.plot(X, Ymeanarr, c='r')
```

```
    plt.show()
    UpDownArr=[]
    for i in range(len(Y)):
        if Y[i]>Ymean:
            UpDownArr.append(1)
        else:
            UpDownArr.append(0)
    #Alpha Helix
    print('Alpha helix:')
    count=0
    for i in range(len(Y)-7):
        arr1=[1, 1, 0, 0, 1, 1, 0, 0]
        arr2=[0, 0, 1, 1, 0, 0, 1, 1]

arr=[UpDownArr[i],UpDownArr[i+1],UpDownArr[i+2],UpDownArr[i+3],
UpDownArr[i+4], UpDownArr[i+5], UpDownArr[i+6], UpDownArr[i+7]]
        if (arr==arr1 or arr==arr2):
            print('Position  {}  to  {}  -'.format(i+1,  i+8),
np.array(arr))
            count+=1
    if count==0:
        print('There are not alpha helix in this sequence')
    #Beta sheet
    print('Beta Strands:')
    count=0
    amparr=[]
    for i in range(len(Y)-5):
        arr1=[1, 0, 1, 0, 1, 0]
        arr2=[0, 1, 0, 1, 0, 1]

arr=[UpDownArr[i],UpDownArr[i+1],UpDownArr[i+2],UpDownArr[i+3],
UpDownArr[i+4], UpDownArr[i+5]]
        if (arr==arr1 or arr==arr2):
            print('Position  {}  to  {}  -'.format(i+1,  i+8),
np.array(arr))
            count+=1
            amparr.append([Y[i],Y[i+1],Y[i+2],Y[i+3],   Y[i+4],
Y[i+5]])
    if count==0:
        print('There are not beta strands in this sequence')
    #amphipaticity
    above=[0 for i in range(count)]
    below=[0 for i in range(count)]
    for i in range(count):
        for j in range(len(amparr[i])):
            if amparr[i][j]>Ymean:
                above[i]+=amparr[i][j]
            else:
                below[i]+=amparr[i][j]
        above[i]=above[i]/3
        below[i]=below[i]/3
```

```
    for i in range(count):
        print('{:.2f}'.format(abs(above[i]-below[i])))

if __name__=="__main__":
    file=open('Q2.fasta')
    #file=open('input.txt')
    data=file.read().splitlines()
    file.close()

a=''
strings=[]
head=[]
for i in data:
    if i[0]=='>':
        head.append(i)
        strings.append(a)
        a=''
    else:
        a+=str(i)
strings.append(a)
strings.pop(0)
for i in range(len(strings)):
    strings[i]=strings[i].replace('\n', '')
for i in range(len(head)):
    head[i]=head[i].replace('\n','')

for i in range(len(strings)):
    amphipaticityplot(strings[i])
```

Output:

```
Alpha helix:
Position 213 to 220 - [0 0 1 1 0 0 1 1]
Position 241 to 248 - [1 1 0 0 1 1 0 0]
Position 243 to 250 - [0 0 1 1 0 0 1 1]
Position 245 to 252 - [1 1 0 0 1 1 0 0]
Beta Strands:
Position 70 to 77 - [0 1 0 1 0 1]
Position 71 to 78 - [1 0 1 0 1 0]
Position 72 to 79 - [0 1 0 1 0 1]
Position 110 to 117 - [0 1 0 1 0 1]
Position 159 to 166 - [0 1 0 1 0 1]
Position 220 to 227 - [1 0 1 0 1 0]
Position 221 to 228 - [0 1 0 1 0 1]
Amphiphatic index of beta sheet no. 1 is 1.87
Amphiphatic index of beta sheet no. 2 is 1.29
Amphiphatic index of beta sheet no. 3 is 1.20
Amphiphatic index of beta sheet no. 4 is 2.21
Amphiphatic index of beta sheet no. 5 is 0.91
Amphiphatic index of beta sheet no. 6 is 1.22
Amphiphatic index of beta sheet no. 7 is 0.98
```



3. For finding the hydrophobic index with provided window length, we plot the normal hydrophobicity plot for the first n and last n elements and for the middle ones, we take the hydrophobicity value of element 'I' as the mean of the previous and next n elements and then plot by using the position of the amino acid sequence as X axis and the hydrophobic index (with window length) as Y axis.

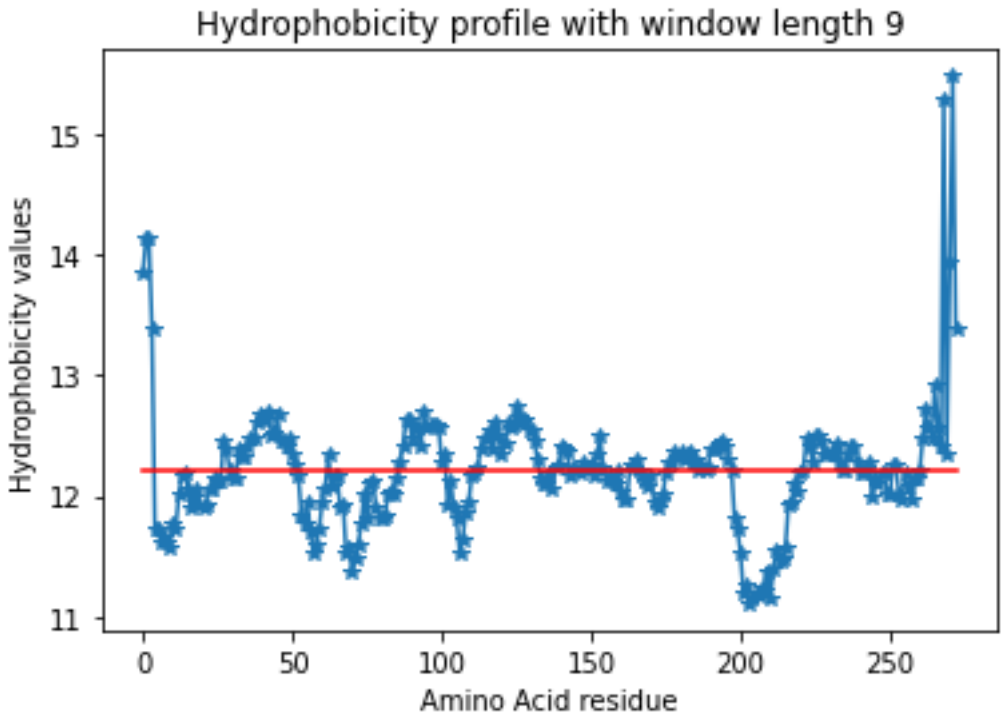
   The python code for plotting the hydrophobic index with a given window length is given below:

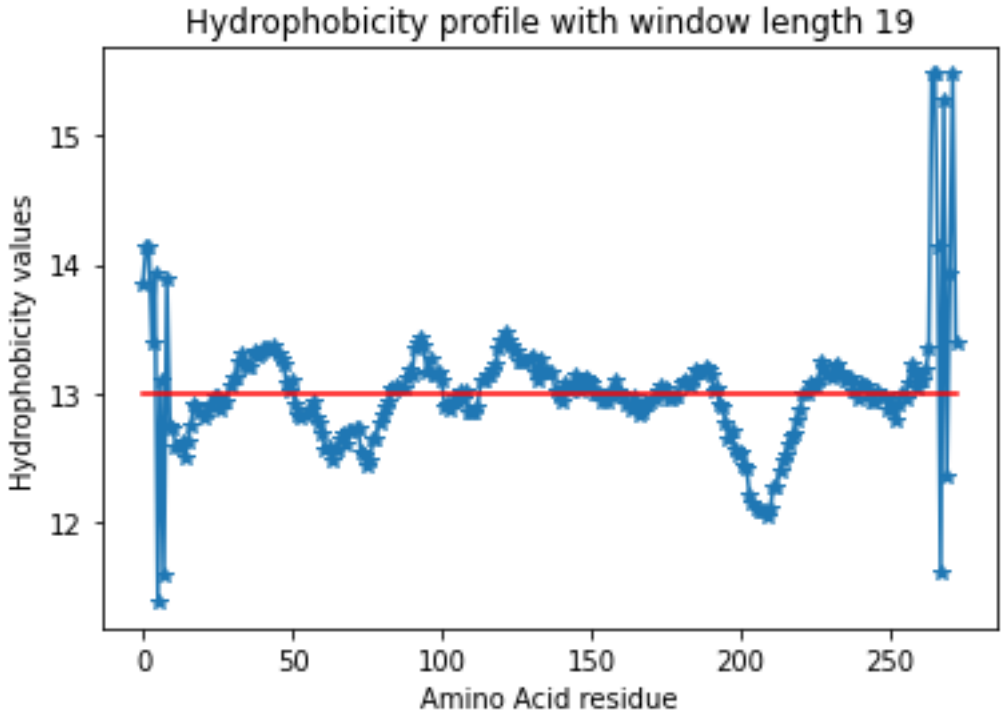
```
import matplotlib.pyplot as plt
import numpy as np
def Hgm_with_window_length(seq, d):
    hydrophobicity_values={'A': 13.85, 'D': 11.61, 'C': 15.37,
'E': 11.38, 'F': 13.93, 'G': 13.34,
                           'H': 13.82, 'I': 15.28, 'K': 11.58,
'L': 14.13, 'M': 13.86, 'N': 13.02,
                           'P': 12.35, 'Q': 12.61, 'R': 13.10,
'S': 13.39, 'T': 12.70, 'V': 14.56,
                           'W': 15.48, 'Y': 13.88}
    H_val = []
    l = int(d/2)
    n = len(seq)
    for i in range(l):
        H_val.append(hydrophobicity_values[seq[i]])
    for i in range(l,n-l-1):
        a=0
        for j in range(-l,l):
            a+=hydrophobicity_values[seq[i+j]]
        a=a/d
        H_val.append(a)
```

```
    for i in range(n-l-1,n):
        H_val.append(hydrophobicity_values[seq[i]])
    plt.plot(H_val,'-*')
    plt.xlabel('Amino Acid residue')
    plt.ylabel('Hydrophobicity values')
    plt.title('Hydrophobicity profile with window length')
    plt.plot(np.mean(H_val)*np.ones(n),'r')
    plt.show()
```

```
a
='ALLSFERKYRVRGGTLIGGDLFDFWVGPYFVGFFGVSAIFFIFLGVSLIGYAASQGPTWD
PFAISINPPDLKYGLGAAPLLEGGFWQAITVCALGAFISWMLREVEISRKLGIGWHVPLAFC
VPIFMFCVLQVFRPLLLGSWGHAFPYGILSHLDWVNNFGYQYLNWHYNPGHMSSVSFLFVNA
MALGLHGGLILSVANPGDGDKVKTAEHENQYFRDVVGYSIGALSIHRLGLFLASNIFLTGAF
GTIASGPFWTRGWPEWWGWWLDIPFWS'
Hgm_with_window_length(a,9)
Hgm_with_window_length(a,19)
```

Output:





4. Using the ScanProsite tool, the number of matches for the given regular expressions are:
   i. [SV]-T-[VT]-[DERK] = 1826
   ii. [FILV]Qxxx{RK}Gxxx[RK]xx[FILVWY] = 40367724
   More details about the patterns are given in the spreadsheet file (
5. The python program for the given question is:

```
def case1(head,string):
    a=['I','L']
    b=['D','E','R','K']
    print(head)
    strlen=6
    strarr=[]
    for i in range(len(string)-strlen+1):
        if  string[i:i+3]=='STV'  and  string[i+3]  in  a  and
string[i+4] in a and string[i+5] not in b:
            print(i, string[i:i+strlen])
            strarr.append(string[i:i+strlen])
    if len(strarr)==0:
           print('No Matches Found')
           #print('')

def case2(head,string):
    c=['F','I','L','V']
    d=['R','K']
    e=['F','I','L','V','W','Y']
    print(head)
    strlen=14
    strarr=[]
    for i in range(len(string)-strlen+1):
```

```
            if string[i] in c and string[i+1]=='Q' and string[i+5]
in d and string[i+6]=='G' and string[i+10] in d and string[i+13]
in e:
                print(i, string[i:i+strlen])
                strarr.append(string[i:i+strlen])
        if len(strarr)==0:
            print('No Matches Found')
            #print('')

if __name__=="__main__":
    file=open('Q4.fasta')
    #file=open('input.txt')
    data=file.read().splitlines()
    file.close()

a=''
string=[]
head=[]
for i in data:
    if i[0]=='>':
        head.append(i)
        string.append(a)
        a=''
    else:
        a+=str(i)
string.append(a)
string.pop(0)
for i in range(len(string)):
    string[i]=string[i].replace('\n', '')
for i in range(len(head)):
    head[i]=head[i].replace('\n','')

for i in range(len(head)):
    print('Search result for the pattern S-T-V-[IL](2)-{DERK}')
    print('position and pattern in the given sequence:')
    case1(head[i],string[i])
for j in range(len(head)):
    print('Search result for the pattern [FILV]-Q-x-x-x-[RK]-G-
x-x-x-[RK]-x-x-[FILVWY]')
    print('position and pattern in the given sequence:')
    case2(head[i],string[i])
```

The Output of the program is given in a separate txt file (file name: A8_Q5_BE19B029_Out.txt)

6. The answer is given in the text files: (A8_Q6_SMSPFF.txt and A8_Q6_SPT.txt)