Question 1

In [58]:

```
import string
import pandas as pd
import numpy as np
from collections import Counter
aas = sorted(set(string.ascii_uppercase) - set("BJOZXU"))
def numerator(s,p):
    c = Counter()
    for x,y in zip(s,p):
        if y == 'H':
            if x in c:
                c[x] += 1
            else:
                c[x] = 1
    c =  {x:y / s.count(x) for x,y in c.items()}
    for a in aas:
        if a not in c:
            c[a] = 0
    return c
def denom(p):
    return p.count('H') / len(p)

def prop(s,p):
    d = {x : y / denom(p) for x,y in numerator(s,p).items()}
    df = pd.DataFrame(data = [d.values()]).T
    df.index = d.keys()
    df.columns = ["Propensity"]
    return df
```

In [59]:

```
s = "LGASGIAAFAFGSTAILIILFNMAAEVHFDPLQFFRQFFWLGLYPPKAQYGMGIPPLHDGGWWLMAGLFMTLSLGSWWIRVYSRARALGLGTHIAWNFAAAIFFVLCIG
p = "XHHHHHHHHHHHHHHHHHHHHHHHHHXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXHHHHHHHHHHHHHHHHHHHHHHHHHHHHHXXHHHHHHHHHHHHHHHH
prop(s,p)
```

Out[59]:

	Propensity
G	1.061224
A	1.551020
S	1.360544
I	1.200480
F	1.020408
T	0.765306
L	1.224490
N	1.360544
M	1.530612
E	0.408163
W	1.113173
R	0.680272
V	0.583090
Y	0.583090
H	0.874636
C	1.020408
P	0.226757
D	0.000000
K	0.000000
Q	0.000000

Question 2

Amino Acid	In Helix	In Seqeunce	N(Helix)	Seq length	Propensity
A	19	25	98	200	1.551020408
C	2	4	98	200	1.020408163
D	0	5	98	200	0
E	1	5	98	200	0.4081632653
F	10	20	98	200	1.020408163
G	13	25	98	200	1.06122449
H	3	7	98	200	0.8746355685
I	10	17	98	200	1.200480192
K	0	1	98	200	0
L	12	20	98	200	1.224489796
M	3	4	98	200	1.530612245
N	2	3	98	200	1.360544218
P	1	9	98	200	0.2267573696
Q	0	4	98	200	0
R	3	9	98	200	0.6802721088
S	6	9	98	200	1.360544218
T	3	8	98	200	0.7653061224
V	2	7	98	200	0.583090379
W	6	11	98	200	1.113172542
Y	2	7	98	200	0.583090379

Question 3

In [3]:

```
pa = {x:y for x,y in zip("EALHMQWVFKIDTSRCNYPG",[1.53,1.45,1.34,1.24,1.20,1.17,1.14,1.14,1.12,1.07,1.00,0.98,0.82,
pb = {x:y for x,y in zip("MVICYFQLTWARGDKSHNPE",[1.67,1.65,1.60,1.30,1.29,1.28,1.23,1.22,1.20,1.19,0.97,0.90,0.81,
npa = {'E': 1, 'A': 1, 'L': 1, 'H': 1, 'M': 1, 'Q': 1, 'W': 1, 'V': 1, 'F': 1, 'K': 0.5, 'I': 0.5, 'D': 0, 'T': 0,
npb = {'M': 1, 'V': 1, 'I': 1, 'C': 1, 'Y': 1, 'F': 1, 'Q': 1, 'L': 1, 'T': 1, 'W': 1, 'A': 0.5, 'R': 0, 'G': 0, '
s = "KVFGRCELAAAMKRHGLDNYRGYSLGNWVCAAKFESNFNTQATNRNTDGSTDYGILQINSRWWCNDGRTPGSRNLCNIPCSALLSSDITASVNCAKKIVSDGNGMNAWV
```

In [4]:

```
helices = []
strands = []
i = 0
while(i < len(s)-5):
    if sum([npa[aa] for aa in s[i:i+6]]) >= 4:
        j = i+2
        while(sum([pa[aa] for aa in s[j:j+4]]) >= 4):
            j += 1
        helices.append(s[i:j+3])
        i = j+3
    i += 1

i = 0
while(i < len(s)-4):
    if sum([npb[aa] for aa in s[i:i+5]]) >= 3:
        j = i+2
        while(sum([pb[aa] for aa in s[j:j+3]]) >= 3):
            j += 1
        strands.append(s[i:j+2])
        i = j+2
    i += 1

hr = []
br = []
for x in helices:
    for y in strands:
        if x in y or y in x:
            seq = sorted([x,y], key = len)[0]
            if sum([pa[aa] for aa in seq]) > sum([pb[aa] for aa in seq]):
                br.append(y)
            else:
                hr.append(x)
df = pd.DataFrame([set(helices)-set(hr), set(strands)-set(br)]).T
df.columns = ["Helices","Strands"]
df
```

Out[4]:

	Helices	Strands
0	RCELAAAMKRH	TDYGILQIN
1	MNAWVAWRN	GTDVQAWIRGCRL
2	None	WVCAA
3	None	VFGRC

Question 4

A) Alpha Helix

1. Checking if simplified propensitites of a 6 segement residue sums up, atleast to 4

Sequence	Breakdown	Sum	
KVFGRC	0.5 + 1 + 1 - 1 + 0 + 0	1.5	
VFGRCE	1 + 1 - 1 + 0 + 0 + 1	2	
FGRCEL	1 - 1 + 0 + 0 + 1 + 1	2	
GRCELA	-1 + 0 + 0 + 1 + 1 + 1	2	
RCELLA	0 + 0 + 1 + 1 + 1 + 1	4	Condition satisfied

2. Extending the sequence

Sequence	Breakdown	Sum	
ELAA	1.53 + 1.34 + 1.45 + 1.45	5.77	
LAAM	1.34 + 1.45 + 1.45 + 1.20	5.69	
AAMK	1.45 + 1.45 + 1.20 + 1.07	5.55	
AMKR	1.45 + 1.20 + 1.07 + 0.79	4.51	
MKRH	1.20 + 1.07 + 0.79 + 1.24	4.3	
KRHG	1.07 + 0.79 + 1.24 + 0.53	3.64	Stop Extending

3. Alpha helix = RCELAAMKRH

B) Beta Strand			
1. Checking if simplified propensitites of a 5 segement residue sums up, atleast to 3			
Sequence	**Breakdown**	**Sum**	
KVFGR	-1 + 1 + 1 + 0 + 0	1	
VFGRC	1 + 1 + 0 + 0 + 1	3	Condition Satisfied
2. Extending the sequence			
Sequence	**Breakdown**	**Sum**	
GRC	0.81 + 0.9 + 1.3	3.01	
RCE	0.9 + 1.3 + 0.26	2.46	Stop Extending
3. Beta Strand = VFGRC			