Problem 4 (a)

First, import the **math** library

In [1]:
```
import math
import scipy.special
```

Then move on to the formal math

Define R.V.s

We define the random variables X and Y as

- X - The outcome of a series
- Y - The number of games in a series

The R.V. X has the probabilities

- **A** wins in 5 : $(0.5)^5 = 0.03125$ with multiplicity 1
- **A** wins in 6 : $(0.5)^5(0.5)^1 = 0.015625$ with multiplicity $\binom{5}{4}$
- **A** wins in 7 : $(0.5)^5(0.5)^2 = 0.0078125$ with multiplicity $\binom{6}{4}$
- **A** wins in 8 : $(0.5)^5(0.5)^3 = 0.00390625$ with multiplicity $\binom{7}{4}$
- **A** wins in 9 : $(0.5)^5(0.5)^4 = 0.00195313$ with multiplicity $\binom{8}{4}$
- **B** wins in 5 : $(0.5)^5 = 0.03125$ with multiplicity 1
- **B** wins in 6 : $(0.5)^5(0.5)^1 = 0.015625$ with multiplicity $\binom{5}{4}$
- **B** wins in 7 : $(0.5)^5(0.5)^2 = 0.0078125$ with multiplicity $\binom{6}{4}$
- **B** wins in 8 : $(0.5)^5(0.5)^3 = 0.00390625$ with multiplicity $\binom{7}{4}$
- **B** wins in 9 : $(0.5)^5(0.5)^4 = 0.00195313$ with multiplicity $\binom{8}{4}$

Thus, we define the probability and multiplicity arrays for X

In [2]:
```
probX = []
probX.append((0.5)**5)
probX.append(((0.5)**5)*((0.5)**1))
probX.append(((0.5)**5)*((0.5)**2))
probX.append(((0.5)**5)*((0.5)**3))
probX.append(((0.5)**5)*((0.5)**4))
probX.append((0.5)**5)
probX.append(((0.5)**5)*((0.5)**1))
probX.append(((0.5)**5)*((0.5)**2))
probX.append(((0.5)**5)*((0.5)**3))
probX.append(((0.5)**5)*((0.5)**4))

print(probX)
print(len(probX))
```

```
[0.03125, 0.015625, 0.0078125, 0.00390625, 0.001953125, 0.03125, 0.015625, 0.007
8125, 0.00390625, 0.001953125]
10
```

In [3]:
```
multX = []
multX.append(1)
multX.append(scipy.special.binom(5, 4))
multX.append(scipy.special.binom(6, 4))
multX.append(scipy.special.binom(7, 4))
multX.append(scipy.special.binom(8, 4))
multX.append(1)
multX.append(scipy.special.binom(5, 4))
multX.append(scipy.special.binom(6, 4))
multX.append(scipy.special.binom(7, 4))
multX.append(scipy.special.binom(8, 4))

print(multX)
print(len(multX))
```

```
[1, 5.0, 15.0, 35.0, 70.0, 1, 5.0, 15.0, 35.0, 70.0]
10
```

Now, the R.V. Y has the probabilities

- Series lasts 5 : $2(0.5)^5 = 0.0625$
- Series lasts 6 : $2\binom{5}{4}(0.5)^5(0.5)^1 = 0.15625$
- Series lasts 7 : $2\binom{5}{4}(0.5)^5(0.5)^2 = 0.234375$
- Series lasts 8 : $2\binom{5}{4}(0.5)^5(0.5)^3 = 0.273438$
- Series lasts 9 : $2\binom{5}{4}(0.5)^5(0.5)^4 = 0.273438$

Thus, we define the probability array for Y

In [4]:
```
probY = []
probY.append(2*probX[0]*multX[0])
probY.append(2*probX[1]*multX[1])
probY.append(2*probX[2]*multX[2])
probY.append(2*probX[3]*multX[3])
probY.append(2*probX[4]*multX[4])

print(probY)
print(len(probY))
```

```
[0.0625, 0.15625, 0.234375, 0.2734375, 0.2734375]
5
```

Before computing the entropies $H(x)$ and $H(y)$, we check that the probabilities we calculated are normal

In [5]:
```
xSum = 0
ySum = 0
for i in range(5):
    xSum += 2 * probX[i] * multX[i]
    ySum += probY[i]

print(xSum)
print(ySum)
```

```
1.0
1.0
```

Since the probabilities are normal, we can calculate $H(x)$ as

In [6]:
```
Hx = 0
for i in range(10):
    Hx += multX[i] * (- probX[i] * math.log2( probX[i] ))

print(Hx)
```

```
7.5390625
```

and $H(y)$ as

In [7]:
```
Hy = 0
for i in range(5):
    Hy += - probY[i] * math.log2( probY[i] )

print(Hy)
```

```
2.18206960194
```

Now, since $p(x, y)$ is

In [8]:
```
probXY = []
for i in range(10):
    xyRow = []
    for j in range(5):
        if i == j:
            xyRow.append(1/2)
        elif i % 5 == j:
            xyRow.append(1/2)
        else:
            xyRow.append(0)

    probXY.append(xyRow)


print(probXY)
```

```
[[0.5, 0, 0, 0, 0], [0, 0.5, 0, 0, 0], [0, 0, 0.5, 0, 0], [0, 0, 0, 0.5, 0], [0,
0, 0, 0, 0.5], [0.5, 0, 0, 0, 0], [0, 0.5, 0, 0, 0], [0, 0, 0.5, 0, 0], [0, 0, 0
, 0.5, 0], [0, 0, 0, 0, 0.5]]
```

and $H(x, y)$ as

In [9]:
```
Hxy = 0
for i in range(10):
    for j in range(5):
        tmpArr = probXY[i]
        tmp = tmpArr[j]

        if tmp == 0:
            Hxy = Hxy
        else:
            Hxy += - tmp * math.log2(tmp)


print(Hxy)
```

```
5.0
```

We also have the conditional probability $H(x \mid y)$ as

In [10]:
```
HxGIVy = 0
for i in range(5):
    # From player A winning in (i+5) games
    tmpA = multX[i] * (- (probX[i]*probY[i]) * math.log2(probX[i]*probY[i]))

    # From player B winning in (i+5) games
    tmpB = multX[i+5] * (- (probX[i+5]*probY[i]) * math.log2(probX[i]*probY[i]))

    tmpTOT = tmpA + tmpB
    HxGIVy += tmpTOT

print(HxGIVy)
```

```
2.29731956998
```

As well as the conditional probability $H(y \mid x)$ as

In [11]:
```
HyGIVx = 0
for i in range(10):
    HyGIVx += multX[i] * (- probY[i % 5]*probX[i] * math.log2(probY[i % 5]*probX[i
]))

print(HyGIVx)
```

```
2.29731956998
```

Using the relation $I(X;Y) = H(X) + H(Y) - H(X,Y)$, we have $I(X;Y)$ as

In [12]:
```
Ixy = Hx + Hy - Hxy
print(Ixy)
```

```
4.72113210194
```

Next, we need to define the distributions pA (for X when A wins) and qA (for Y when A wins). The distribution pA can be represented by the matrix

In [13]:
```
pA = []
for i in range(5):
    tmp = multX[i] * (probX[i] + probX[i+5])
    pA.append(tmp)

print(pA)
```

```
[0.0625, 0.15625, 0.234375, 0.2734375, 0.2734375]
```

Similarly, the distribution qA can be represented by the matrix

In [14]:
```
qA = []
for i in range(5):
    qA.append(probY[i])

print(qA)
```

```
[0.0625, 0.15625, 0.234375, 0.2734375, 0.2734375]
```

After checking these new distributions for normality, we proceed.

In [15]:
```
pAsum = 0
qAsum = 0
for i in range(5):
    pAsum += pA[i]
    qAsum += qA[i]

print(pAsum)
print(qAsum)
```

```
1.0
1.0
```

For $D(pA \mid\mid X)$ we have

In [16]:
```
dPaX = 0
for i in range(5):
    dPaX += pA[i] * math.log2(pA[i] / probX[i])

print(dPaX)
```

```
5.35699289806
```

In [17]:
```
dPaX = 0
for i in range(5):
    dPaX += pA[i] * math.log2(pA[i] / (multX[i]*probX[i]))

print(dPaX)
```

```
1.0
```

and for $D(qA \mid\mid Y)$ we have

In [18]:
```
dQaY = 0
for i in range(5):
    dQaY += qA[i] * math.log2(qA[i] / probY[i])

print(dQaY)
```

```
0.0
```