

<b>J.P.Morgan Quant Mentorship Program 2023</b>
---

<b>Name: Tanisha Salvi</b>
----------------------------

<b>Institute: IIT Guwahati</b>
--------------------------------

<b>Branch: Computer Science and Engineering</b>
---

<b>Year: B.Tech(2021-2025), Second year</b>
---

## **PART A**

### **1(a)**

100 wheat contracts of long or short hedges will be needed.

There are 500,000 bushels of wheat in total, with each future contract of 5,000 bushels of wheat. Here, this is an example of short hedge.

$$\frac{500000}{5000} = 100 \text{ contracts}$$

### **1(b)**

Future contracts predetermine the price for which the farmer will sell the wheat. If the market value of wheat falls below the price at which the future contracts are sold, the farmer profits since he receives higher future price. If the market value of wheat rises above the price at which future contracts are sold, the farmer incurs a loss as he has to now sell wheat at a lower price compared to the market price. However, this loss will be offset by the gain from the futures.

## **2.**

There are three contracts being opened and closed, each of these contracts are short contracts which are being sold at a higher value than what they are being bought at.

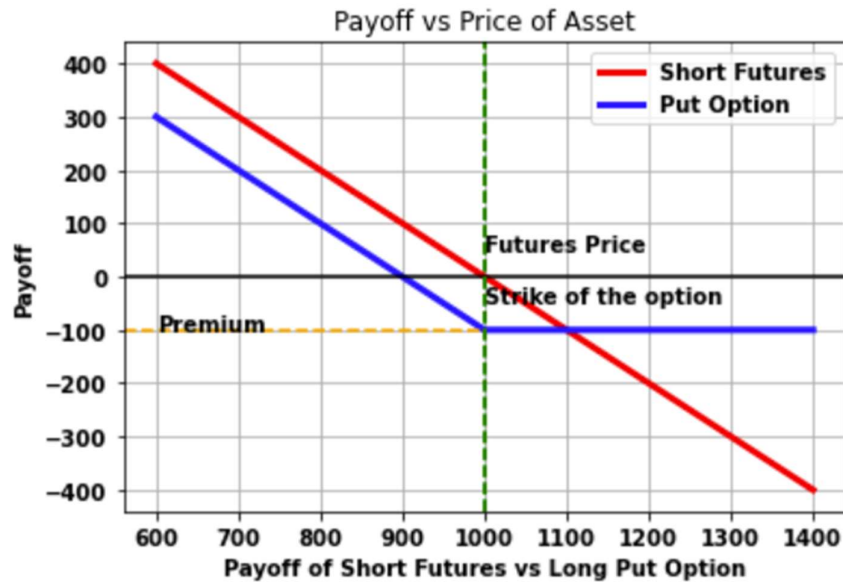
Thus, the company incurs profit on each of these contracts-

<b>Contract</b>	<b>Bought at</b>	<b>Sold at</b>	<b>Profit</b>
October futures	119.20	119.80	0.60
March futures	118.00	118.80	0.80
July futures	117.10	117.60	0.50

Thus, total net profit = 0.60 + 0.80 + 0.50

= 1.90 (per barrel of oil)

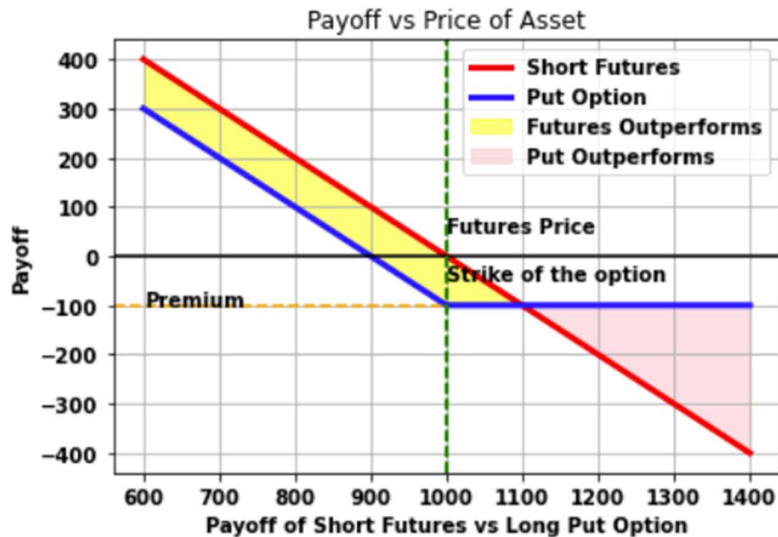
3.



The red line corresponds to the payoff due to short future contract (Trader X), whereas the blue one is for payoff due to put option (Trader Y).

Trader X has an agreement to sell the asset at the fixed price (Future price = \$1000), at a fixed time (1 year) in future. If the price of asset goes down, he will buy it at a lower price, and sell at \$1000, thus incurring profit. However, if the price goes up, he will be at loss.

Trader Y has a right, but not obligation to sell the asset at predetermined price (\$1000) at the same fixed time (1 year) in future. If price of asset goes down, he can sell the asset for \$1000 and make profit, however, he has to pay a premium of \$100. It is due to this premium that the payoff line is shifted downwards by the value of premium. The major difference from Trader X comes in the case when price of asset goes up, Trader Y has the option to not use his option, and he loses only the premium amount \$100.



The code can be found at - [Tanisha\\_Salvi\\_Case\\_Study\\_Part\\_A\\_Q\\_3\\_MAIN.py](#)

Payoffs corresponding to both strategies are plotted. It can be observed that the payoff line for Trader X (short futures) is above that for Trader Y (pay option) up to the price asset value = \$1100. Hence, future contract outperforms put option up to asset price = \$1100.

Above that, both strategies are in losses, however, the loss due to pay option remains constant at \$100, but loss for future strategy goes increasing ( $\geq \$100$ ). Thus, in this region, put option outperforms the future contract.

## **PART B**

**a) Call Option** - the trader has the right, but no obligation to buy the asset anytime up to the expiration date at the strike price itself. Let the premium paid by trader be  $Prem(C)$ .

**Put Option** - the trader has the right, but no obligation to sell the asset anytime up to the expiration date at the strike price itself. Let the premium paid by trader be  $Prem(P)$ .

Maximum Payoffs -

Call Option =  $\max(0, \text{spot price} - \text{strike price}) - Prem(C)$

$$\text{Put Option} = \max (0, \text{strike price} - \text{spot price}) - \text{Prem(P)}$$

Since the trader has both options, the options being in/out of the money will depend on the spot price and strike price values.

Spot Price > Strike Price	Call -> ITM and Put -> OTM
Spot Price < Strike Price	Call -> OTM and Put -> ITM

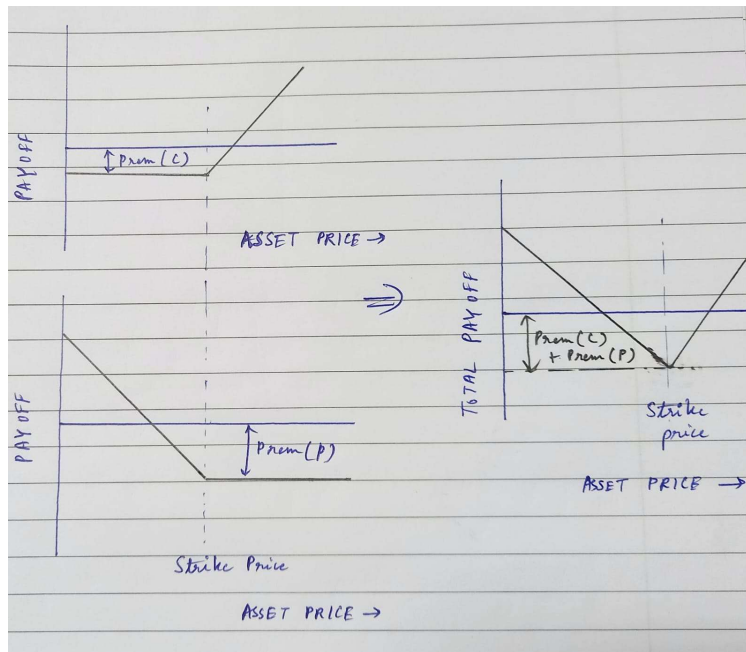
ITM – In the money

OTM – Out of the money

This is a “synthetic long” situation, as either of Call or Put options is in the money, the other will automatically be out of the money.

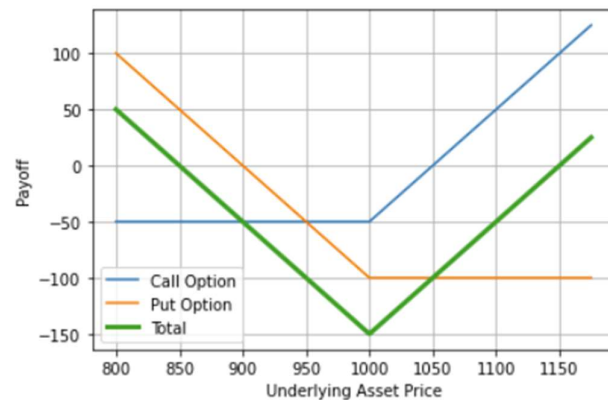
Note – If Spot Price = Strike Price, both payoff options are worthless.

$$\begin{aligned} \text{Net Payoff} = & \max (0, \text{spot price} - \text{strike price}) - \text{Prem(C)} \\ & + \max (0, \text{strike price} - \text{spot price}) - \text{Prem(P)} \end{aligned}$$



An example taking arbitrary values for  $Prem(C)$  and  $Prem(P)$  is plotted using matplotlib.

The code can be found at - [Tanisha\\_Salvi\\_Case\\_Study\\_Part\\_B\\_Q\\_a\\_MAIN.py](#)



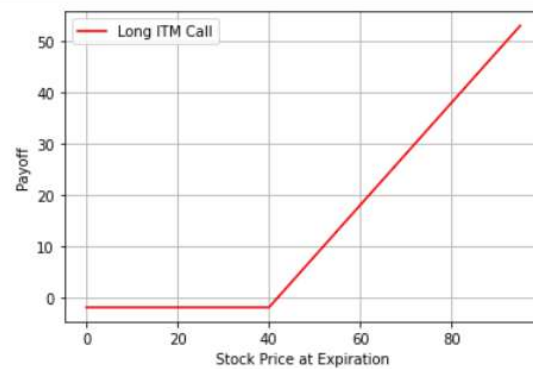
- b) ITM - In the money  
 OTM - Out of the money  
 ATM = At the money

The trader has exercised three variations of options simultaneously.  
 A look at the discussion for each of them -

### Long ITM Call Option

The payoff for ITM Call Option will be positive if  
 $\text{Spot Price} > \text{Strike Price}$ .

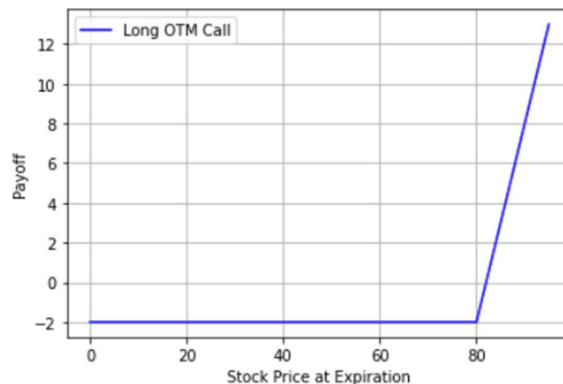
This ITM call thus has an intrinsic value.



### Long OTM Call Option

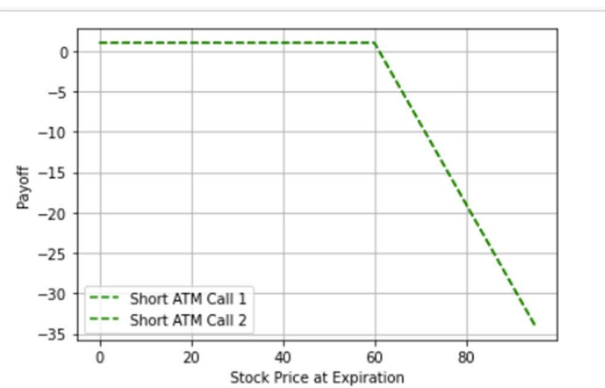
The payoff for OTM Call Option will be zero if  
 $\text{Spot Price} < \text{Strike Price}$

This OTM call thus does not have an intrinsic value.



### Short ATM Call Option

There are two short ATM Call options offset the intrinsic value of the long ITM call option, due to the same distance between each strike price of constituent leg. The trader will hence have the payoff that only depends on the difference in intrinsic value between ITM and OTM call options.



The above three graphs are constructed without including premium.

Now, the final graph (combination of all four options) will be shifted by net premium value.

Let

premium for long ITM call =  $p_1$

premium for short ATM call =  $p_2$

premium for long OTM call =  $p_3$

The trader will pay premiums  $p_1$  and  $p_3$ , and receive premium  $2 * p_2$  (note there are two short ATM call options).

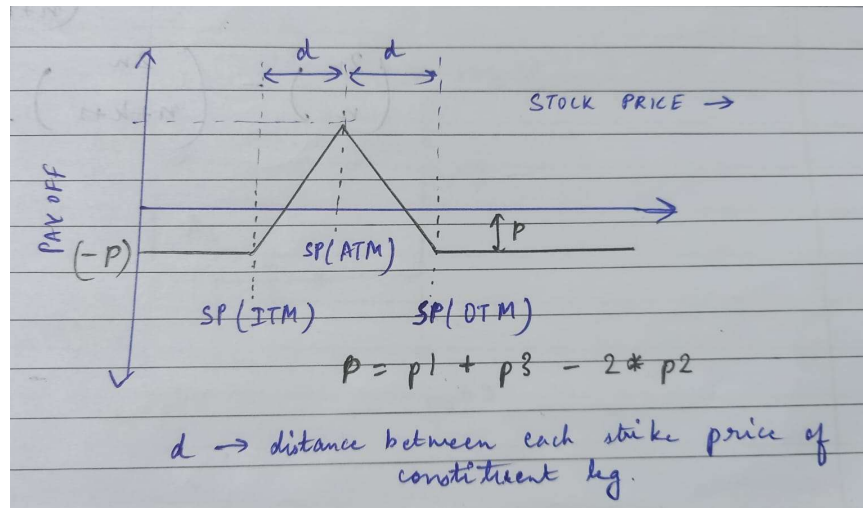
Thus, the net premium paid by trader =  $p_1 + p_3 - 2 * p_2$

On combining the four options, for net payoff, the payoff line lies above X-axis. The net premium has not been taken into consideration as of now. Now, this is an interesting observation, that the net premium that we have to pay has to be positive (so that the payoff line shifts down), because if so is not the case, then this strategy implies that we can make money from practically 0 amount of money, which is arbitrage and exists very rarely in financial markets.

Thus,  $p_1 + p_3 - 2 * p_2 > 0$  (this is the net premium paid by trader).

The net graph shifts below X - axis by this net value.





Here,

SP(ITM) = Strike price for ITM Call

SP(ATM) = Strike price for ATM Call

SP(OTM) = Strike price for OTM Call

Net premium P (to be paid) =  $p_1 + p_3 - 2 * p_2$

In the region before SP(ITM), no call is exercised. Between SP(ITM) and SP(ATM), trader gets profit from ITM Option. Between SP(ATM) and SP(OTM), net payoff begins decreasing due to ATM calls simultaneously being exercised. Further beyond SP(OTM), all three calls are exercised simultaneously. Let current Strike Price be  $SP(OTM) + x$ .

Net Payoff =  $(SP(OTM) - SP(ITM) + x) - 2 * (SP(OTM) - SP(ATM) + x) + x$

Let  $SP(OTM) - SP(ATM) = d$ , then  $SP(OTM) - SP(ITM) = 2d$  [difference between each strike price of constituent leg is same]

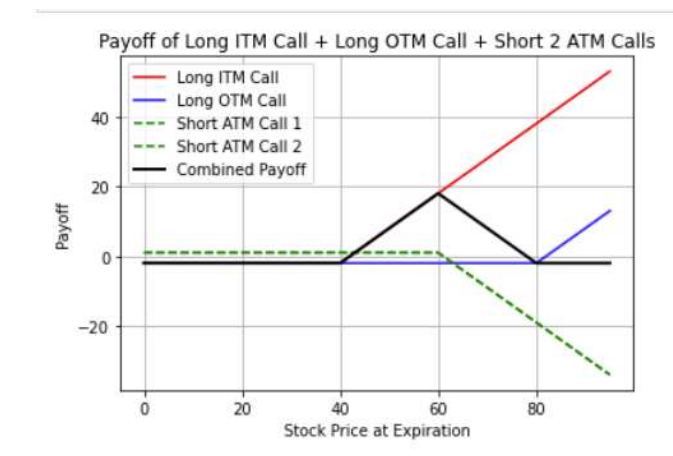
Therefore, Net Payoff =  $(2d + x) - 2(d + x) + x = 0$

[Note, premium is not included in these calculations]

An example, corresponding to 1 long ITM call, 1 long OTM call, 2 short ATM calls, with

Option	Strike Price
Long ITM Call	40
Short ATM Call	60
Long OTM Call	80

Note that the question states the distance between each strike price of constituent leg is same. (In the example here, taken to be 20). Also, the net premium is not considered during graph construction. The graph will be shifted according to the value of net premium.



The code file for the above graph is -

**Tanisha\_Salvi\_Case\_Study\_Part\_B\_Q\_b\_MAIN.py**

- c) ITM - In the money  
OTM - Out of the money  
ATM = At the money

The trader has two Options-

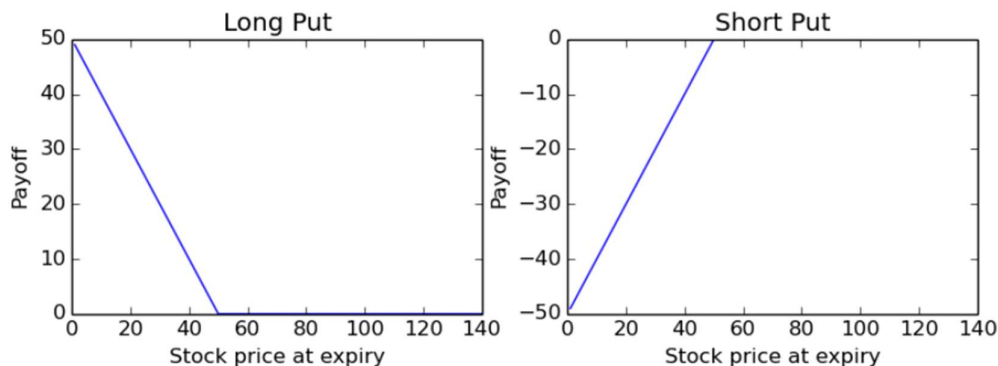
#### Buying 1 ITM Put Option

This is a long-put option. The trader can sell the asset at the strike price of the option which is higher than the spot price. The trader will have a loss due to the increase in the price of asset. This strategy can be named "bullish" as the trader benefits due to the increase in the asset price.

### Selling 1 OTM Put Option

This is a short-put option. Here, the trader has the obligation to sell the asset at the strike price, which is lower than the spot price, to benefit. Hence, it is a “bearish” strategy, such that the trader profits due to decrease in the asset price.

General representations on how both these strategies work -



These graphs can be shifted down according to the premium. Note that these are payoffs plotted at expiry.

Both strategies can be combined, this combination of Long ITM Put and Short OTM Put will lead to the variation in payoff for the trader.

This combination of “bullish and bearish” strategy results will result in the net payoff for the trader, which will depend on the variation in the underlying asset value.

Since, both the expiration date and underlying assets are same, the combined payoff will depend on the strike prices of the two options.

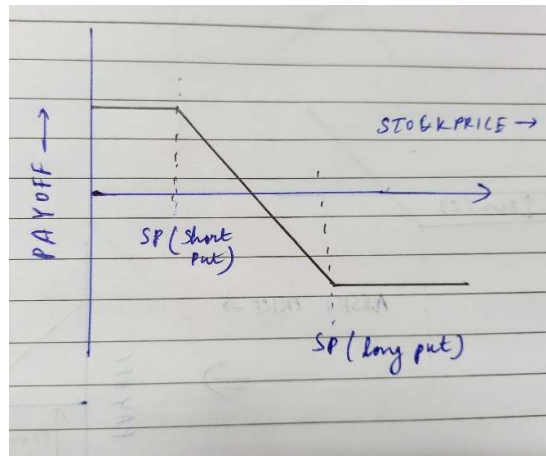
Note that since both are put options, the Strike Price (OTM) will be lesser than Strike Price(ITM).

Before Strike Price(OTM), both ITM and OTM Put options are exercised.

Between Strike Price(OTM) and Strike Price(ITM), ITM is exercised.

After Strike Price(ITM), none of the options are exercised.

The final graph will look like -



## PROBLEM 1

### 1.1

$$(x, y) = \lim_{i \rightarrow \infty} (x_i, y_i) \quad , \quad (x_0, y_0) = (0, 0)$$

$(x, y)$  lies in the first quadrant.

At each step, the ant can move in either +x or +y direction. Also, the ant begins at  $(0, 0)$ . This clearly means the ant can never move leftwards crossing y-axis or downwards crossing x-axis.

### 1.2

$$\begin{aligned} d_m &= |x - 0| + |y - 0| \\ &= |x| + |y| = x + y \end{aligned}$$

{From Q1.2, it is found that  $(x, y)$  lies in the first quadrant, hence both  $x \geq 0$  and  $y \geq 0$ . Thus,  $|x| = x$  and  $|y| = y$ }

Now,  $E(d_m) = E(x + y)$

An observation on the steps of ant -

In either step, it moves either in +x or +y direction, so there is some movement in each step (in either direction). Whether the current increase happens in  $x_i$  or in  $y_i$ , the overall increase always occurs in  $x_i + y_i$ . The length of each step decreases by a factor of 2 from previous, beginning with 1. These step lengths form a Geometric Progression with ratio = 0.5

$$1, \frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \dots$$

$(x_i + y_i)$  increases by an additional factor of step length in each move, i.e., by the geometric progression number.

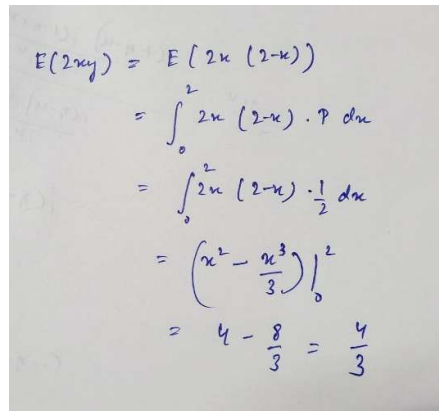
$$\begin{aligned} (x + y) &= \lim_{i \rightarrow \infty} (x_i + y_i) \\ E(d_m) = E(x + y) &= \sum_{i=0}^{\infty} \frac{1}{2^i} \\ &= \frac{1}{1 - 1/2} = 2 \end{aligned}$$

### 1.3

$$\begin{aligned} d_e &= \sqrt{(y_1 - y_2)^2 + (x_1 - x_2)^2} \\ &= \sqrt{(y - 0)^2 + (x - 0)^2} \\ &= \sqrt{(y)^2 + (x)^2} \end{aligned}$$

$$\begin{aligned} E(d_e^2) &= E(y^2 + x^2) = E((x + y)^2 - 2xy) \\ &= (2)^2 - E(2xy) \end{aligned}$$

$$E(2xy) = E(2x(2 - x)) \quad [\text{replacing } y = 2 - x]$$



Handwritten calculation of  $E(2xy)$  using integration:

$$\begin{aligned} E(2xy) &= E(2x(2-x)) \\ &= \int_0^2 2x(2-x) \cdot p \, dx \\ &= \int_0^2 2x(2-x) \cdot \frac{1}{2} \, dx \\ &= \left( x^2 - \frac{x^3}{3} \right) \Big|_0^2 \\ &= 4 - \frac{8}{3} = \frac{4}{3} \end{aligned}$$

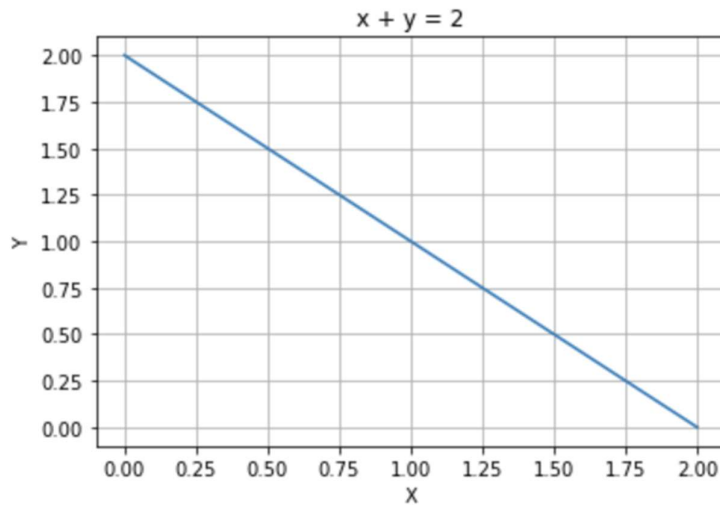
$$\begin{aligned} E(d_e^2) &= 4 - E(2xy) = 4 - \frac{4}{3} \\ &= \frac{8}{3} \end{aligned}$$

Note - Here, the distribution is taken to be nearly continuous while calculating  $E(2xy)$  as every point on  $x + y = 2$  [within boundary points  $(0,2)$  and  $(2,0)$ ] is reachable as  $i$  tends to infinity.

### 1.4

The equation of the locus representing all possible values of  $(x,y)$  will be a straight line with slope -1, only in the first quadrant, i.e., between  $(2,0)$  and  $(0,2)$ .

$$x + y = 2$$



The ant always moves in either +x or +y direction. Ultimately after infinite steps, when it reaches  $(x,y)$ , we find that the sum of  $E(x+y) = 2$  is a constant. This can be interpreted as the Manhattan distance of point  $(x,y)$  is constant from start point  $(0,0)$ . No matter whatever values  $x$  and  $y$  can take, for any point  $(x,y)$ ,  $x + y = 2$ . Thus, the locus of point

$$(x,y) = \lim_{i \rightarrow \infty} (x_i, y_i)$$

Will be line  $x + y = 2$ .

Note - The locus is not the entire line  $x + y = 2$ , but the part of line that lies within the first quadrant, extending from  $(2,0)$  to  $(0,2)$  as represented clearly in the above figure.

## PROBLEM 2

### 2.1

The total number of configurations for a deck of 'n' cards will be given by  ${}^{2n}C_n$  where 2n is the total number of cards (both black and red).

For our possible configurations of 8 cards, total number of configurations

$$\begin{aligned} &= \binom{8}{4} \\ &= 8! / (4! * 4!) \\ &= 70 \end{aligned}$$

### 2.2

The code for generating all possible configurations can be found in the file - **Tanisha\_Salvi\_Case\_Study\_Problem\_2\_Q\_2.2\_MAIN.py**

The expected payoff for this game, as per calculations in code  
= **1.3285714285714285**

The output corresponding to the code in the above file -

Total cards = 4 with 4 red and 4 black cards.

```
Total number of configurations : 70
['R', 'R', 'R', 'R', 'B', 'B', 'B', 'B'] -> 4
['R', 'R', 'R', 'B', 'R', 'B', 'B', 'B'] -> 3
['R', 'R', 'R', 'B', 'B', 'R', 'B', 'B'] -> 3
['R', 'R', 'R', 'B', 'B', 'B', 'R', 'B'] -> 3
['R', 'R', 'R', 'B', 'B', 'B', 'B', 'R'] -> 3
['R', 'R', 'B', 'R', 'R', 'B', 'B', 'B'] -> 3
['R', 'R', 'B', 'R', 'B', 'R', 'B', 'B'] -> 2
['R', 'R', 'B', 'R', 'B', 'B', 'R', 'B'] -> 2
['R', 'R', 'B', 'R', 'B', 'B', 'B', 'R'] -> 2
['R', 'R', 'B', 'B', 'R', 'R', 'B', 'B'] -> 2
['R', 'R', 'B', 'B', 'R', 'B', 'R', 'B'] -> 2
['R', 'R', 'B', 'B', 'B', 'R', 'R', 'B'] -> 2
['R', 'R', 'B', 'B', 'B', 'R', 'B', 'R'] -> 2
['R', 'R', 'B', 'B', 'B', 'B', 'R', 'R'] -> 2
['R', 'B', 'R', 'R', 'R', 'B', 'B', 'B'] -> 3
['R', 'B', 'R', 'R', 'B', 'R', 'B', 'B'] -> 2
['R', 'B', 'R', 'R', 'B', 'B', 'R', 'B'] -> 2
['R', 'B', 'R', 'R', 'B', 'B', 'B', 'R'] -> 2
['R', 'B', 'R', 'B', 'R', 'R', 'B', 'B'] -> 2
```



```

['R', 'B', 'R', 'B', 'R', 'B', 'R', 'B'] -> 1
['R', 'B', 'R', 'B', 'R', 'B', 'B', 'R'] -> 1
['R', 'B', 'R', 'B', 'B', 'R', 'R', 'B'] -> 1
['R', 'B', 'R', 'B', 'B', 'R', 'B', 'R'] -> 1
['R', 'B', 'R', 'B', 'B', 'B', 'R', 'R'] -> 1
['R', 'B', 'B', 'R', 'R', 'R', 'B', 'B'] -> 2
['R', 'B', 'B', 'R', 'R', 'B', 'R', 'B'] -> 1
['R', 'B', 'B', 'R', 'R', 'B', 'B', 'R'] -> 1
['R', 'B', 'B', 'R', 'B', 'R', 'R', 'B'] -> 1
['R', 'B', 'B', 'R', 'B', 'R', 'B', 'R'] -> 1
['R', 'B', 'B', 'R', 'B', 'B', 'R', 'R'] -> 1
['R', 'B', 'B', 'B', 'R', 'R', 'R', 'B'] -> 1
['R', 'B', 'B', 'B', 'R', 'R', 'B', 'R'] -> 1
['R', 'B', 'B', 'B', 'R', 'B', 'R', 'R'] -> 1
['R', 'B', 'B', 'B', 'B', 'R', 'R', 'R'] -> 1
['B', 'R', 'R', 'R', 'R', 'B', 'B', 'B'] -> 3
['B', 'R', 'R', 'R', 'B', 'R', 'B', 'B'] -> 2
['B', 'R', 'R', 'R', 'B', 'B', 'R', 'B'] -> 2
['B', 'R', 'R', 'R', 'B', 'B', 'B', 'R'] -> 2
['B', 'R', 'R', 'B', 'R', 'R', 'B', 'B'] -> 2
['B', 'R', 'R', 'B', 'R', 'B', 'R', 'B'] -> 1
['B', 'R', 'R', 'B', 'R', 'B', 'B', 'R'] -> 1
['B', 'R', 'R', 'B', 'B', 'R', 'R', 'B'] -> 1
['B', 'R', 'R', 'B', 'B', 'R', 'B', 'R'] -> 1
['B', 'R', 'R', 'B', 'B', 'B', 'R', 'R'] -> 1
['B', 'R', 'B', 'R', 'R', 'R', 'B', 'B'] -> 2
['B', 'R', 'B', 'R', 'R', 'B', 'R', 'B'] -> 1
['B', 'R', 'B', 'R', 'R', 'B', 'B', 'R'] -> 1
['B', 'R', 'B', 'R', 'B', 'R', 'R', 'B'] -> 1
['B', 'R', 'B', 'R', 'B', 'R', 'B', 'R'] -> 0
['B', 'R', 'B', 'R', 'B', 'B', 'R', 'R'] -> 0
['B', 'R', 'B', 'B', 'B', 'R', 'R', 'R'] -> 1
['B', 'R', 'B', 'B', 'B', 'R', 'R', 'B'] -> 0
['B', 'R', 'B', 'B', 'B', 'R', 'B', 'R'] -> 0
['B', 'R', 'B', 'B', 'B', 'B', 'R', 'R'] -> 0
['B', 'B', 'R', 'R', 'R', 'R', 'B', 'B'] -> 2
['B', 'B', 'R', 'R', 'R', 'B', 'R', 'B'] -> 1
['B', 'B', 'R', 'R', 'R', 'B', 'B', 'R'] -> 1
['B', 'B', 'R', 'R', 'B', 'R', 'R', 'B'] -> 1
['B', 'B', 'R', 'R', 'B', 'R', 'B', 'R'] -> 0
['B', 'B', 'R', 'R', 'B', 'B', 'R', 'R'] -> 0
['B', 'B', 'R', 'B', 'R', 'R', 'R', 'B'] -> 1
['B', 'B', 'R', 'B', 'R', 'R', 'B', 'R'] -> 0
['B', 'B', 'R', 'B', 'R', 'B', 'R', 'R'] -> 0
['B', 'B', 'R', 'B', 'B', 'R', 'R', 'R'] -> 0
['B', 'B', 'B', 'R', 'R', 'R', 'R', 'B'] -> 1
['B', 'B', 'B', 'R', 'R', 'R', 'B', 'R'] -> 0
['B', 'B', 'B', 'R', 'R', 'R', 'B', 'R'] -> 0
['B', 'B', 'B', 'R', 'R', 'B', 'R', 'R'] -> 0
['B', 'B', 'B', 'R', 'B', 'R', 'R', 'R'] -> 0
['B', 'B', 'B', 'B', 'R', 'R', 'R', 'R'] -> 0

```

The expected payoff for possible stated configurations = 1.3285714285714285

### 2.3

The possible values of payoff if we play the game with standard 52 card deck is a set containing all integers from 0 to 26 (including both)

{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26}

### 2.4

We need to calculate the number of configurations for which payoff is  $\leq 0$ . This is same as calculating the number of configurations for which payoff = 0.

(This can be concluded from the fact that the player plays 'optimally,' that is, tries to maximize his/her payoff for each configuration. The total number of cards is 8, with 4 red and 4 black cards, thus even if the player draws out all cards, his/her payoff will be 0 [+4 due to 4 red cards + (-4) due to 4 black cards]. When a player can get a payoff of 0 in any configuration, he/she will always prefer that draw over any other draw which results in negative payoff.)

Hence,  $n_0$ : number of deck configurations, where max payoff = 0

In all such configurations, a pattern can be observed. If we place the cards in their draw order, and count the number of red ( $n_R$ ) and number of black ( $n_B$ ) cards up to  $i^{\text{th}}$  draw ( $1 \leq i \leq 2n$ ),

This can be proven from the fact that if till any 'i' draws,  $n_R > n_B$  we will have payoff  $> 0$  for that configuration. Hence, the maximum payoff for that configuration cannot be equal to 0. (The player has liberty to stop at any stage, he/she plays optimally.)

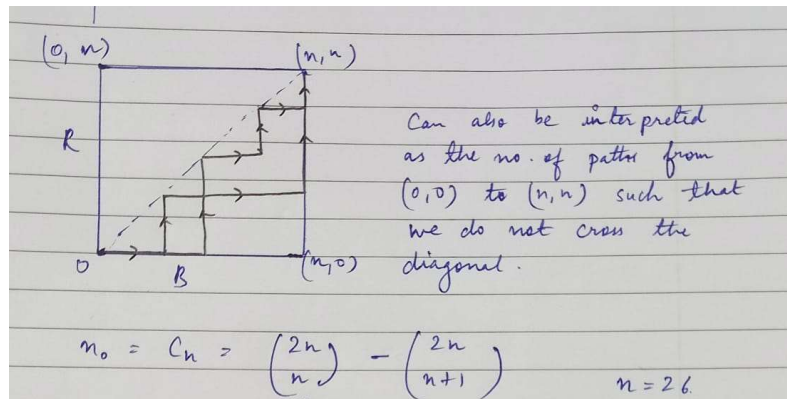
This can be observed as valid bracket sequence, where the number of opening brackets is greater than or equal to the number of closing brackets, also total number of closing brackets = total number of opening brackets.

This problem can be solved using 'Catalan Numbers.'

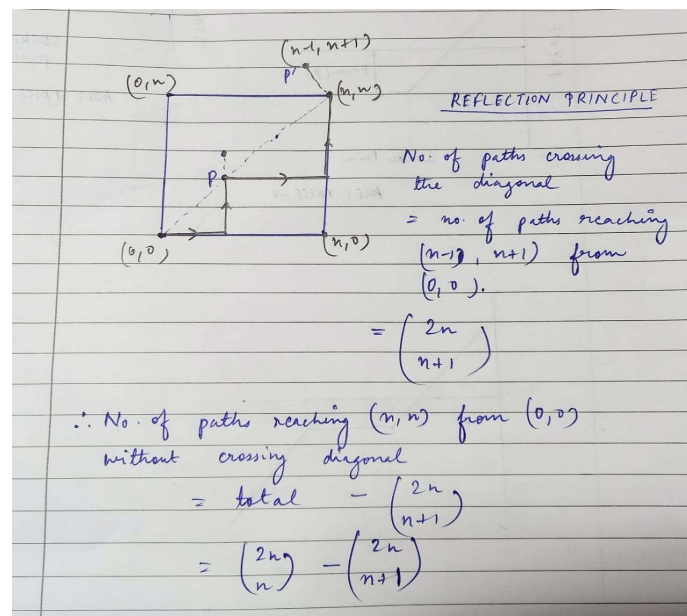
$$\begin{aligned} n_0 = C_n &= \binom{2n}{n} - \binom{2n}{n+1} \\ &= \frac{1}{n+1} \binom{2n}{n} \end{aligned}$$

Reflection principle can also be used -

Here, consider the number of paths from  $(0,0)$  to  $(n,n)$  such that we do not cross the diagonal.

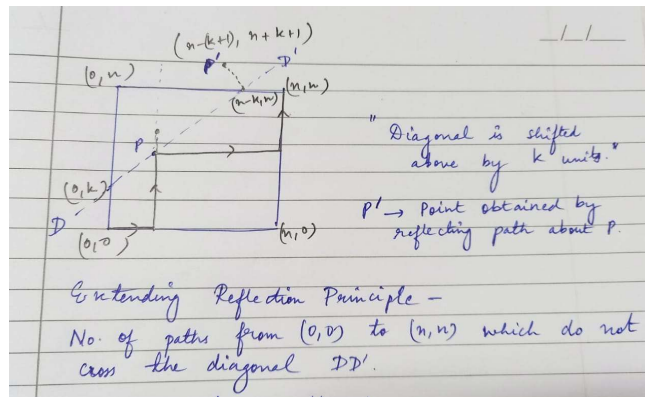


Here,  $P'$  is the point obtained by reflecting the path from  $P$  to  $(n,n)$  about diagonal.  $P' = (n-1, n+1)$



## 2.5

Extending the Reflection Principle to find the number of configurations such that the payoff  $\leq k$ , shifting the diagonal by  $k$  units above, then finding the number of paths from  $(0,0)$  to  $(n,n)$  such that we do not cross this diagonal.



Here,  $P'$  is the point obtained by reflecting the path from  $P$  to  $(n,n)$  about diagonal  $DD'$ .  $P' = (n-(k+1), n+(k+1))$

$$\begin{aligned}
 \text{No. of paths which cross the diagonal, will} \\
 &= \binom{2n}{n+k+1} \\
 &= \text{no. of paths reaching } (n-(k+1), n+(k+1)) \\
 &\quad \text{from } (0,0) \\
 \therefore \text{No. of paths reaching } (n,n) \text{ from } (0,0) \\
 &\quad \text{without crossing diagonal} \\
 &= \text{total} - \binom{2n}{n+k+1} \\
 &= \binom{2n}{n} - \binom{2n}{n+k+1}
 \end{aligned}$$

$$n_k = \binom{2n}{n} - \binom{2n}{n+k+1}$$

Where  $n_k$  is the number of configurations such that the payoff  $\leq k$ .

## 2.6

Cumulative Frequency Distribution function (CDF) of a probability distribution contains probabilities that a random variable  $X$  is less than or equal to  $X$ .

Here, CDF for possible payoffs will be probability that the payoff is less than or equal to  $i$ , where  $0 \leq i \leq 26$  (the possible payoffs)

Using, result given in Q 2.5,

P (payoff  $\leq k$ )

$$\begin{aligned} &= \frac{n_k}{\binom{2n}{n}} = \frac{\binom{2n}{n} - \binom{2n}{n+k+1}}{\binom{2n}{n}} = 1 - \frac{\binom{2n}{n+k+1}}{\binom{2n}{n}} \\ &= 1 - \frac{n!n!}{(n+k+1)!(n-k-1)!} \end{aligned}$$

## 2.7

$f_k$  = the probability that the maximum possible pay-off is greater than or equal to  $k$

$$\begin{aligned} &= P(\text{payoff} \geq k) \\ &= 1 - P(\text{payoff} < k) \\ &= 1 - P(\text{payoff} \leq k-1) \end{aligned}$$

Using result derived in Q2.6,

$$f_k = 1 - \left(1 - \frac{n!n!}{(n+k)!(n-k)!}\right) = \frac{n!n!}{(n+k)!(n-k)!}$$

$$f_{k-1} = 1 - \left(1 - \frac{n!n!}{(n+k-1)!(n-(k-1))!}\right) = \frac{n!n!}{(n+k-1)!(n-k+1)!}$$

Thus,

$$\begin{aligned} \frac{f_k}{f_{k-1}} &= \frac{(n+k-1)!(n-k+1)!}{(n+k)!(n-k)!} \\ &= \frac{n-k+1}{n+k} \end{aligned} \quad \begin{array}{l} \text{where } 1 \leq k \leq 26, \\ f_0 = 1 \end{array}$$

This expression will help in calculating the values of  $f_k$  without calculating large factorials, since we already know the value of  $f_0$ ,

$$f_1 = \frac{n}{n+1} f_0$$

$$f_2 = \frac{n+1}{n+2} f_1$$

and so on.

## 2.8

The code snippet for the logic is in the file -  
**Tanisha\_Salvi\_Case\_Study\_Problem\_2\_Q\_2.8\_MAIN.py**

The logic applied is the same as derived in Q2.7, which avoids calculating large factorial values and taking advantage of the calculated  $\frac{f_k}{f_{k-1}}$ .

```
if __name__ == '__main__':
# Total number of cards = 2 * n = 52
    n = 26
    f = [0] * (n + 1)
    prob = [0] * (n + 1)

# fk -> probability that max_payoff >= k
# fk / fk-1 = (n-k+1) / (n+k)
    f[0] = 1
    for i in range(1, n + 1):
        f[i] = (n - i + 1) / (n + i) * f[i-1]

# prob(max_payoff = k) = prob(max_payoff >= k) - prob(max_payoff >= k+1)
    for i in range(0, n):
        prob[i] = f[i] - f[i+1]
        print('P ( payoff =', i, ') = %.15f'%prob[i])

# prob(max_payoff > n) = 0
    prob[n] = f[n] - 0
    print('P ( payoff =', n, ') = %.15f'%prob[n])
```

The output corresponding to the above code snippet is -

---

```
P ( payoff = 0 ) = 0.037037037037037
P ( payoff = 1 ) = 0.103174603174603
P ( payoff = 2 ) = 0.148239372377303
P ( payoff = 3 ) = 0.166028097062580
P ( payoff = 4 ) = 0.158377032681816
P ( payoff = 5 ) = 0.133080701072915
P ( payoff = 6 ) = 0.100085485930870
P ( payoff = 7 ) = 0.067931325292445
P ( payoff = 8 ) = 0.041793939179924
P ( payoff = 9 ) = 0.023355436600546
P ( payoff = 10 ) = 0.011860442199708
P ( payoff = 11 ) = 0.005469477104627
P ( payoff = 12 ) = 0.002286570695914
P ( payoff = 13 ) = 0.000864323723056
P ( payoff = 14 ) = 0.000294354149586
P ( payoff = 15 ) = 0.000089901267361
P ( payoff = 16 ) = 0.000024481740474
P ( payoff = 17 ) = 0.000005901245982
P ( payoff = 18 ) = 0.000001247692008
P ( payoff = 19 ) = 0.000000228719099
P ( payoff = 20 ) = 0.000000035811446
P ( payoff = 21 ) = 0.000000004694793
P ( payoff = 22 ) = 0.000000000501342
P ( payoff = 23 ) = 0.000000000041890
P ( payoff = 24 ) = 0.000000000002569
P ( payoff = 25 ) = 0.000000000000103
P ( payoff = 26 ) = 0.000000000000002
```

$P(\text{payoff} > 26) = 0$

{Argument already stated in Q2.3}

## 2.9

The most likely (one with highest probability) payoff for the standard 52 card deck = 3, with probability = 0.166028097062580

(As seen and compared among values for probabilities for each possible payoff calculated in Q2.8)

## 2.10

The probability array (prob) evaluated in Q2.8 is used here, a function is created and called, the expected payoff is calculated returned.

The entire code file is -

**Tanisha\_Salvi\_Case\_Study\_Problem\_2\_Q\_2.10\_MAIN.py**

Below is a snippet of the function used to calculate the expected payoff -

```
# function to find the expected payoff using the probabilities calculated  
# P(payoff = k) 0 <= k <= 26  
def find_expected_payoff(prob, n):  
    expected_payoff = 0.0  
    #expected payoff is the summation of (probability that payoff = k) * k  
    for i in range(0, n + 1):  
        expected_payoff += i * prob[i]  
    return expected_payoff
```

The expected payoff = 4.040664774713896