J. P. Morgan Quant Mentorship Program 2021

Palak Kothari

B.Tech (Batch 2023) pkothari@iitg.ac.in IIT Guwahati

Mechanical Engineering (Major) - Computer Science & Engineering (Minor)

Contents

1 Case Study A: Options

1.1 Question 1

Let s_1 and p_1 are strike price and premium for Long Call, s_2 and p_2 for short call, s_3 and p_3 for long put and s_4 and p_4 for short put.Let T_x be total payoff at price x, then:
$T_x = [max(x - s_1, 0) - p_1] + [min(s_2 - x, 0) + p_2] + [max(s_3 - x, 0) - p_3] + [max(x - s_4, 0) + p_4]$

1.1.1 Part A

First Derivative: Long a call with Strike = \$50 and Premium = \$4
We have bought a call with Strike \$50 after paying a premium of \$4. As the holder of call option, we have the right to buy the stock in either of the below mentioned ways:

1. **When x $>$ 50**, buy the stock at the strike price i.e. \$50.
   Total Payoff = Market Price - Strike Price - Premium

2. **When x $<=$ 50**, buy the stock at market price by letting call option expire.
   Total Payoff = $-1*$Premium

Therefore, payoff will be:
= max(market price - strike, 0) - premium
= **max(x-50, 0) - 4**

Second Derivative: Short a put with Strike = \$50 and Premium = \$2.
We have sold a put with Strike \$50 after receiving a premium of \$2. As the seller of put option, we will have to sell the stock in either of the below mentioned ways, as per the holder:

1. **x $<$ 50 :** As the broker of put option, he/she will have to buy the stock at strike price in this case.
   Thus, Total Payoff for short put = Market Price - Strike Price + Premium

2. **x $>=$ 50 :** As the broker of the put option he/she will keep the premium as the holder of put option won't sell the stock at price greater than strike price.
   Thus, Total Payoff = Premium.

Therefore, payoff will be:
= min(market price - strike, 0) + premium
= **min(x-50, 0) + 2**

Combining payoff for both option,
Total Payoff: **max(x-50, 0) + min(x-50, 0)** - 2

Codefile: *Palak_Kothari_A_Q1_A_MAIN.py*

Matplotlib library of the python is used for plotting the payoff graph.

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "*python Palak_Kothari_A_Q1_A_MAIN.py*" to run the program.

Source Code

```
def plot_payoff():
    # Import matplotlib module to plot the graph
    import matplotlib.pyplot as plt

    # Create the vectors X and Y
    x = list(range(100))
    long_call=[]
    short_put=[]
    for i in range(len(x)):
        long_call.append(max(i-50, 0)-4)
        short_put.append(min(i-50, 0)+2)
    y=[]
    for i in range(len(x)):
        y.append(long_call[i]+short_put[i])

    # Create the plot
    plt.plot(x,y, color = "red", label = 'Total Payoff')
    plt.plot(x,long_call, linestyle = 'dashed', color = "blue",
         label = 'Payoff For Long Call', alpha = 0.5)
    plt.plot(x, short_put, linestyle = 'dashed', color = "green",
         label = 'Payoff For Short Put', alpha = 0.5)

    # Add a title
    plt.title('Long a Call & Short a Put | Strike = $50')

    # Add X and y Label
    plt.xlabel('Stock Price ($)')
    plt.ylabel('Total Payoff')

    # Add a grid
    plt.grid(alpha=.4,linestyle='--')

    # Add a Legend
    plt.legend(loc ="lower right", prop={"size":12})
    # Show the plot
    plt.show()

if __name__ == "__main__":
    plot_payoff()
```

Plot Of The Payoff:

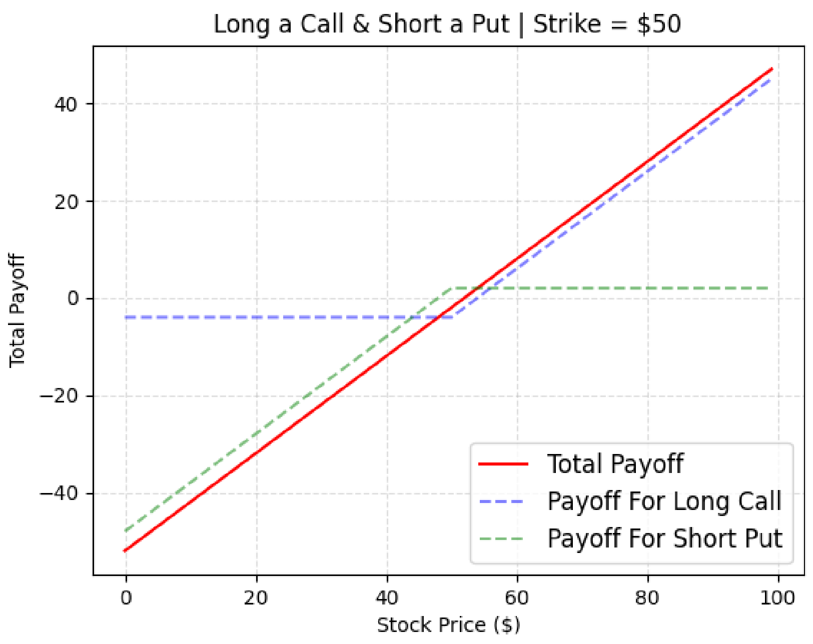


Figure 1: Long Call and Short Put Payoff Plot, Strike Price = 50

1.1.2 Part B

The portfolio is trying to mimic a **Stock**. It resembles with a stock bought at \$50. If the value rises, we make a profit of (stock value - \$50) while if the value of the stock goes down, we incur a loss of (\$50 - stock value). Refer below images for better clarity, Figure 2 is Derivative payoff plot(Red Line) where as Figure 3 is stock payoff plot.

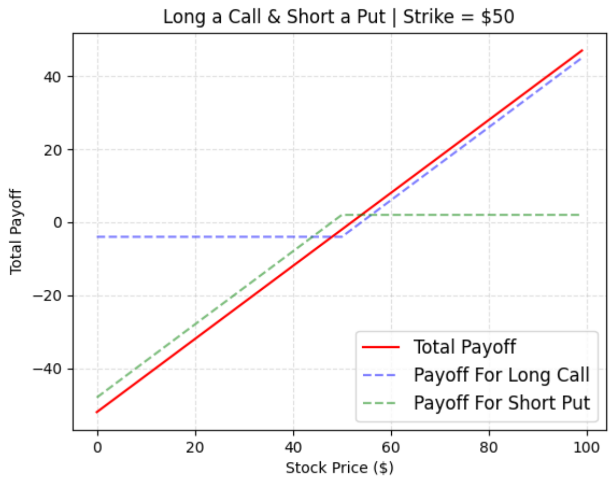


Figure 2: Payoff Plots, Strike Price = 50

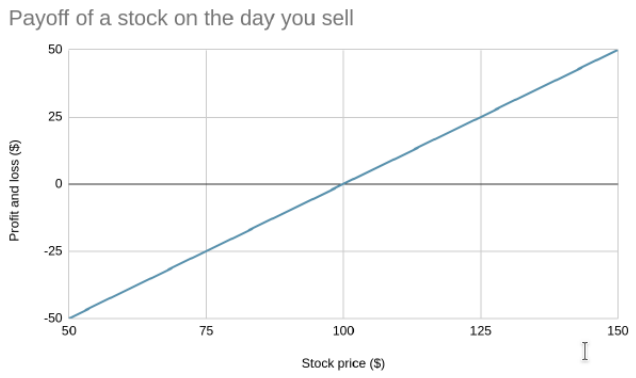


Figure 3: Stock Payoff

1.2 Question 2

1.2.1 Part A

Analysing the graphs of all options at Strike Price = 50.

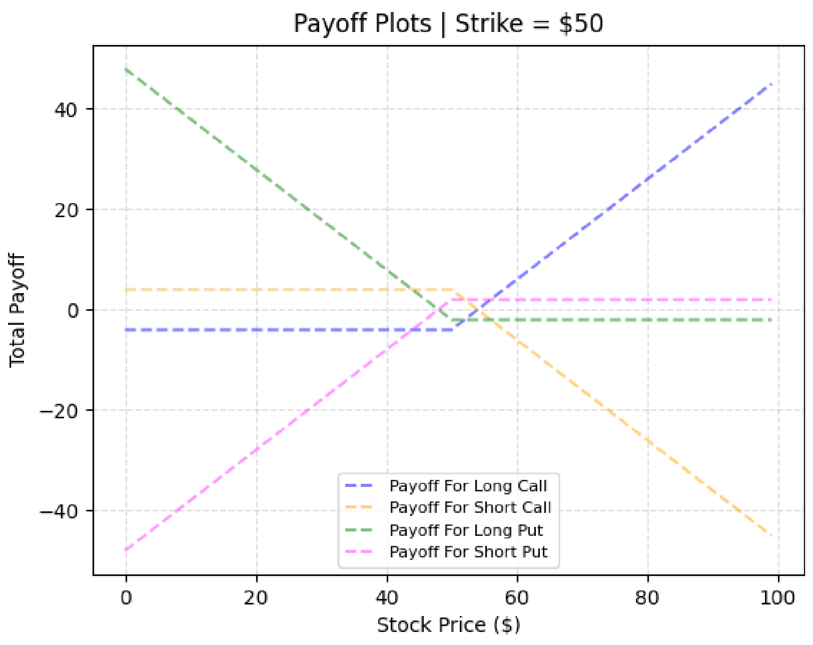


Figure 4: Payoff Plots, Strike Price = 50

As we can see from the plot that a long option nullify corresponding short option, thus taking a long option and it's corresponding short option will not generate required plot. Observing above plot carefully we can see that a **short put(at strike price 50) and a short call(at strike price 50) is minimum requirement to generate the given plot.**

Other combinations can be: Short Put + Short Call + (Long Call + Short Call)* + (Long Put + Short Put)* (all at strike price 50)
$*$ denotes the pair in the parenthesis can be repeated zero or more time.

Codefile: *Palak_Kothari_A_Q2_A_MAIN.py*

Matplotlib library of the python is used for plotting the payoff graph.

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "*python Palak_Kothari_A_Q2_A_MAIN.py*" to run the program.

Source Code:

```
def plot_payoff():

    # Import our modules that we are using
    import matplotlib.pyplot as plt

    # Create the vectors X and Y
    x = list(range(100))
    short_call = []
    short_put=[]
    s2 = 50
    s4 = 50
    p2 = 4
    p4 = 2

    for i in range(len(x)):
        short_call.append(min(s2-i,0)+p2)
        short_put.append(min(i-s4, 0)+p4)
    y=[]
    for i in range(len(x)):
        y.append(short_call[i]+short_put[i])

    # Create the plot
    plt.plot(x,y, color = "red", label = 'Total Payoff')
    plt.plot(x,short_call, linestyle = 'dashed', color = "blue", label =
         'Payoff For Short Call', alpha = 0.5)
    plt.plot(x, short_put, linestyle = 'dashed', color = "orange", label
         = 'Payoff For Short Put', alpha = 0.4)

    # Add a title
    plt.title('Combination of Short Put and Short Call')

    # Add X and y Label
    plt.xlabel('Stock Price ($)')
    plt.ylabel('Total Payoff')

    # Add a grid
    plt.grid(alpha=.4,linestyle='--')

    # Add a Legend
    plt.legend(loc ="lower center", prop={"size":8})

    # Show the plot
    plt.show()

    print("Portfolio Includes:")
    print("A Short Call At Strike Price 50 and A Short Put At Strike
        Price 50")

if __name__ == "__main__":
    plot_payoff()
```

Plot Of The Payoff:

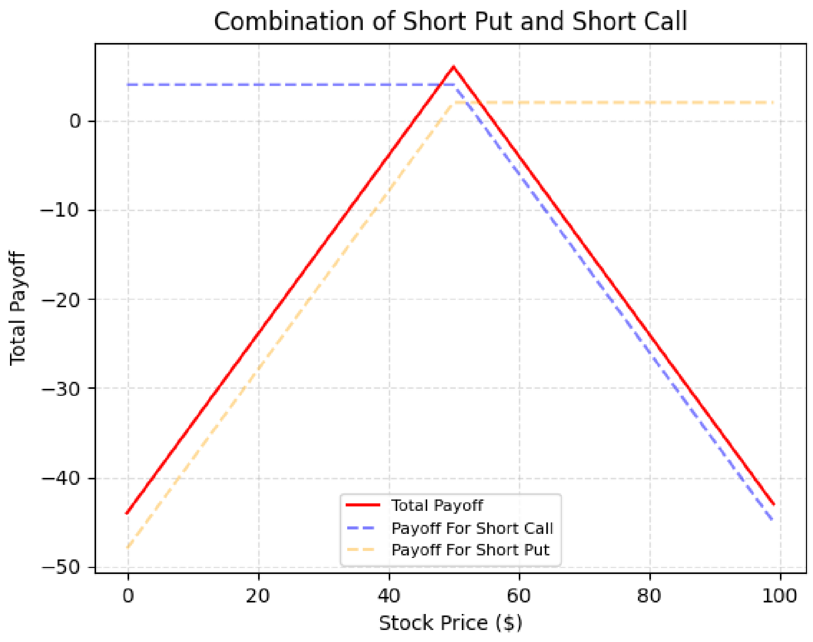


Figure 5: Combination of short put and short call at Strike 50

1.2.2 Part B

As found in the previous part, we can obtain given plot using a short call & short put at the same strike price.(Here, strike price = 50)

This options strategy is popularly known as **Short Straddle**. *In this case, the investor's belief is that the market consensus is too high and considers that the stock price will stay between the breakeven points*(the strike price ± the premium collected).

In this strategy, the maximum profit possible is the option premium collected. Maximum profit happens if the payoff remains in sync with the belief of seller of the options.

While if it goes the other way, the loss is unlimited because the stock price may rise indefinitely on the upside and substantial on the downside(until stock price reaches zero). In this case, the investor makes a loss if the stock price belongs to $[0,44) \cup (56, 100]$.

Hence, the investor believes that the stock price will remain close to the strike price i.e. in between 44 to 56. The maximum profit ($6) is earned if the short straddle is held till expiration and stock price closes exactly at the strike price.

1.3 Question 3

The given payoff can be achieved by buying a call option at a lower strike price, selling two call options struck at the price with highest payoff and buying a call option at a higher strike price.

So, the following needs to be executed to get the given payoff-

- Long a Call(Long Call 1)at strike price = \$50
- Short a Call *two times* at strike price = \$54
- Long a Call (Long Call 2) at strike price = \$58

Note: The premium for call options is \$4 as given in instructions.

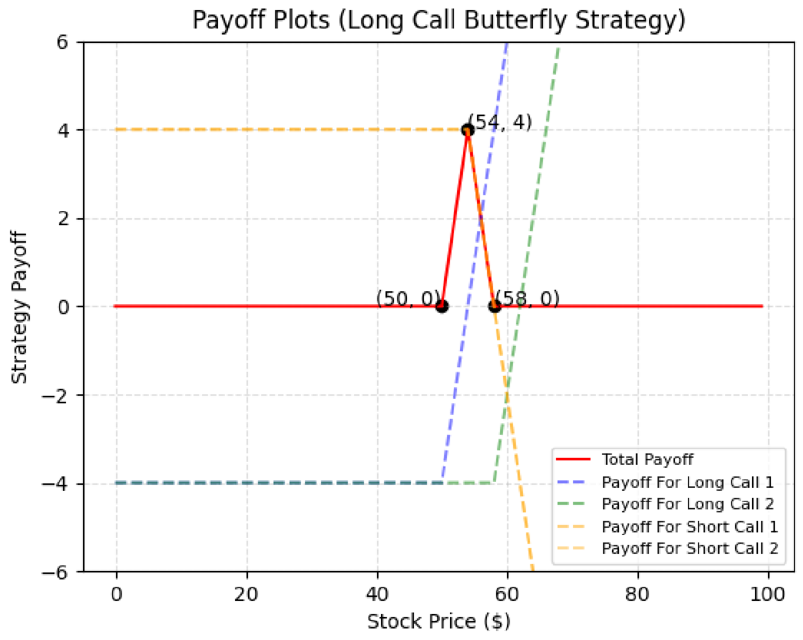


Figure 6: Long Call Butterfly Strategy using 4 Call options

The investor expects that there will not be much movement in Price of the underlying stock and wants that options expire worthless except for the option with lower strike price.

This options trading strategy is known as **Long Call Butterfly** in which the trader expects the stock prices to remain range-bound. It is a *limited risk* and *limited profit* strategy.

In this case, it can be clearly observed from the two payoffs that risks are limited in case of Long Call Butterfly than that of Short Straddle.

Codefile: *Palak_Kothari_A_Q3_MAIN.py*

Matplotlib library of the python is used for plotting the payoff graph.

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "*python Palak_Kothari_A_Q3_MAIN.py*" to run the program.

Source Code:

```
def plot_payoff():

    # Library used for plotting
    import matplotlib.pyplot as plt

    # Create the vectors X and Y
    x = list(range(100))
    long_call1=[]
    short_call1 = []
    long_call2 = []
    short_call2=[]
    s1 = 50
    s2 = 54
    s3 = 58
    s4 = 54
    p1 = 4
    p2 = 4
    p3 = 4
    p4 = 4
    for i in range(len(x)):
        long_call1.append(max(i-s1, 0)-p1)
        short_call1.append(min(s2-i,0)+p2)
        long_call2.append(max(i-s3, 0)-p3)
        short_call2.append(min(s4-i, 0)+p4)
    y=[]
    for i in range(len(x)):
        y.append(short_call1[i]+short_call2[i]+long_call1[i]+long_call2[i])

    axes = plt.gca()
    axes.set_ylim([-6,6])

    # Create the plot
    plt.plot(x,y, color = "red", label = 'Total Payoff')
    plt.plot(x,long_call1, linestyle = 'dashed', color = "blue", label =
        'Payoff For Long Call 1', alpha = 0.5)
    plt.plot(x, long_call2, linestyle = 'dashed', color = "green", label
        = 'Payoff For Long Call 2', alpha = 0.5)
    plt.plot(x,short_call1, linestyle = 'dashed', color = "orange",
        label = 'Payoff For Short Call 1', alpha = 0.5)
    plt.plot(x, short_call2, linestyle = 'dashed', color = "orange",
        label = 'Payoff For Short Call 2', alpha = 0.4)

    # Add a title
    plt.title('Payoff Plots (Long Call Butterfly Strategy)')

    # Add X and y Label
    plt.xlabel('Stock Price ($)')
    plt.ylabel('Strategy Payoff')

    # Add a grid
    plt.grid(alpha=.4,linestyle='--')

    # Add a Legend
    plt.legend(loc ="lower right", prop={"size":8})

    plt.scatter(54,4, color = 'black')
    plt.annotate('({}, {})'.format(54,4), (54,4), horizontalalignment =
        'left')
    plt.scatter(50,0, color = 'black')
    plt.annotate('({}, {})'.format(50,0), (50,0), horizontalalignment =
        'right')
    plt.scatter(58,0, color = 'black')
    plt.annotate('({}, {})'.format(58,0), (58,0), horizontalalignment =
        'left')

    print("Portfolio Includes:")
    print("A Long Call At Strike Price $50 and Premium of $4")
    print("A Long Call At Strike Price $58 and Premium of $4")
    print("Two Short Calls At Strike Price $50 and Premium of $4")

    # Show the plot
    plt.show()

if __name__ == "__main__":
    plot_payoff()
```

1.4 Question 4

Given:

- Calls and puts are available at all strikes in **range** $[\$0, \$100]$ **in multiples of** $\$0.5$ **only**
- The **premiums** of all calls and puts is $\$0.0$.
- Strike of Custom Call Option = $\$50$
- Portfolio Payoff = $\begin{cases} 1, & \text{if } \textit{stock price } > \$50 \\ 0, & \text{if } \textit{stock price } \leq \$50 \end{cases}$

The given portfolio payoff resembles with the payoff of **Digital Call**. A digital call pays 1 if the stock price is above the strike and 0 otherwise.

As we know, call option gives its holder the right to buy the stock at the strike price, irrespective of the market price.

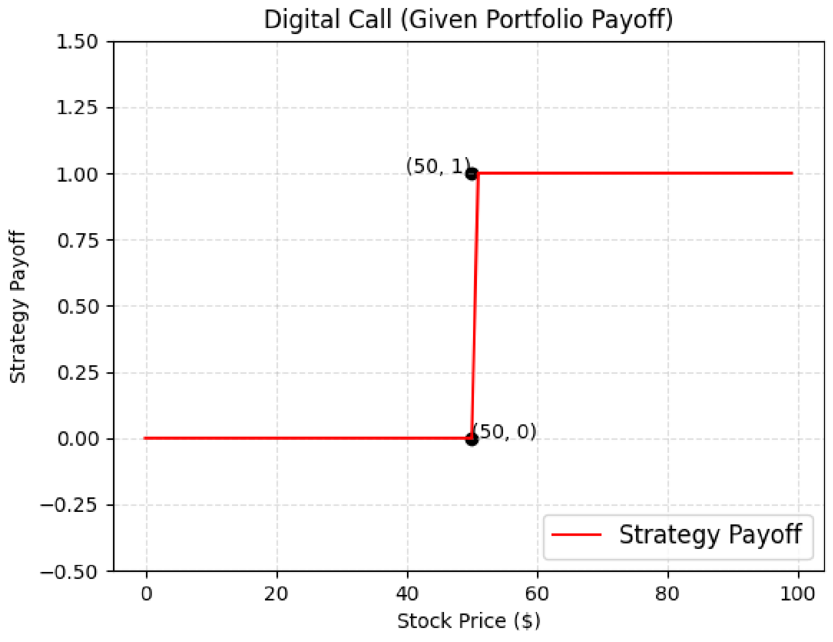


Figure 7: Digital Call (Given Portfolio Payoff)

To create a nearest possible replication of the given portfolio, following call spreads can be adopted-

- For Tighter Lower Bound (Bullish Spread):
  - Long Call with Strike = \$50
  - Short Call with Strike = \$50.5
- For Tighter Upper Bound (Bearish Spread):
  - Long Call with Strike = \$49.5
  - Short Call with Strike = \$50

Considering the first case, if we buy a call at \$50, we can make \$1 for every \$1 appreciation in the stock price which gets cutoff once the stock price reaches \$50.5 as we sold the call at \$50.5, hence cancelling profits from the upside.

But the above mentioned call spreads will give payoff of \$0.5 for stock price $>$ \$50. So, **by doubling the purchase**, we can double the payoff for that particular stock (here, payoff($\textit{stock price} > \$50$) = 1).

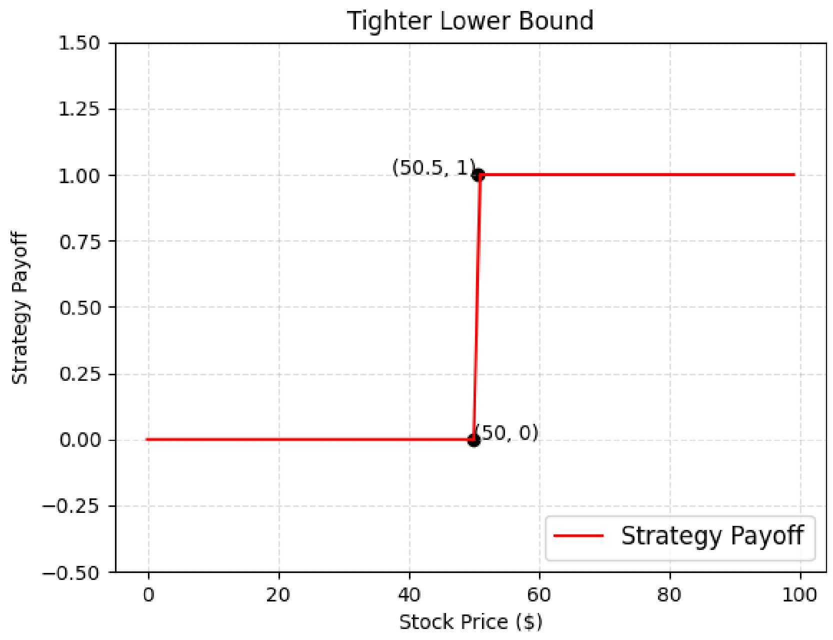


Figure 8: Tighter Lower Bound

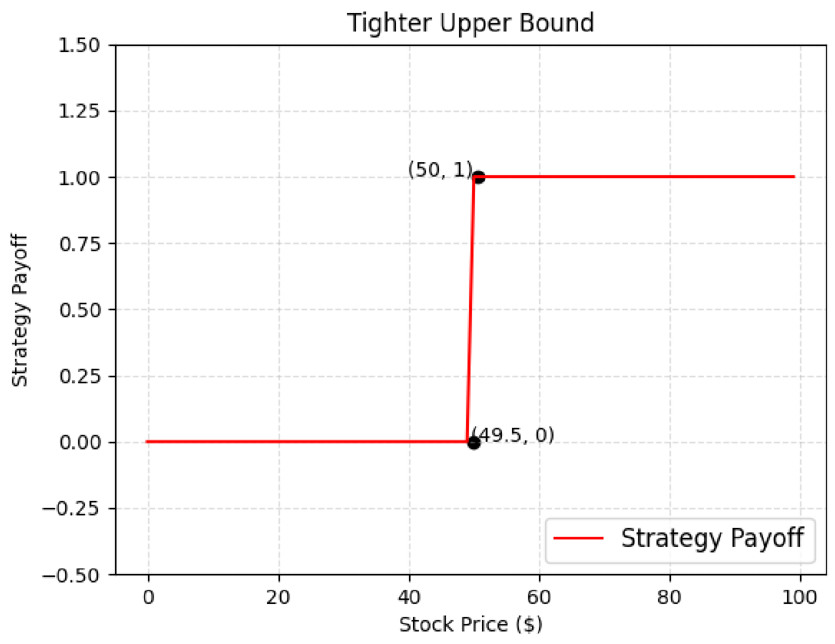


Figure 9: Tighter Upper Bound

Theoretically, the payoff can be exactly replicated by an infinite number of tight call spreads with very small intervals.

Codefile: *Palak_Kothari_A_Q4_LOWER_MAIN.PY*
Palak_Kothari_A_Q4_UPPER_MAIN.PY

Matplotlib library of the python is used for plotting the payoff graph.

Instructions to run the code:

- Open Command Line in the folder containing the code file.
- Run Command "*python Palak_Kothari_A_Q4_LOWER_MAIN.PY*" or "*python Palak_Kothari_A_Q4_UPPER_MAIN.PY*" to run the program.

Source Code for Tighter Upper Bound:

```
def plot_payoff():
    #Python Library for plotting
    import matplotlib.pyplot as plt

    # Create the vectors X and Y
    x = list(range(100))
    long_call=[]
    short_call = []
    lc1 = 49.5
    sc1 = 50
    p1 = 0
    p2 = 0

    for i in range(len(x)):
        long_call.append(max(i-lc1, 0)-p1)
        short_call.append(min(sc1-i,0)+p2)
    y=[]
    for i in range(len(x)):
        y.append(short_call[i]+long_call[i]+short_call[i]+long_call[i])
    axes = plt.gca()
    axes.set_ylim([-0.5,1.5])
    # Create the plot
    plt.plot(x,y, color = "red", label = 'Strategy Payoff')
    # Add a title
    plt.title('Tighter Upper Bound')

    # Add X and y Label
    plt.xlabel('Stock Price ($)')
    plt.ylabel('Strategy Payoff')

    # Add a grid
    plt.grid(alpha=.4,linestyle='--')

    # Add a Legend
    plt.legend(loc ="lower right", prop={"size":12})

    plt.scatter(50,1, color = 'black')
    plt.annotate('({}, {})'.format(50,1), (50,1), horizontalalignment =
         'right')
    plt.scatter(50,0, color = 'black')
    plt.annotate('({}, {})'.format(49.5,0), (49.5,0),
         horizontalalignment = 'left')

    print("Portfolio Includes:")
    print("A Long Call At Strike Price $49.5 and Premium of $0")
    print("A Short Call At Strike Price $50 and Premium of $0")

    # Show the plot
    plt.show()

if __name__ == '__main__':
    plot_payoff()
```

1.5 Question 5

1.5.1 Part A

For Day 1: Let the the price quoted on Day 1 be X and, the factor of increment be **i** ($i >= 1$) and the factor of decrement be **d** ($d < 1$). (See figure 5)

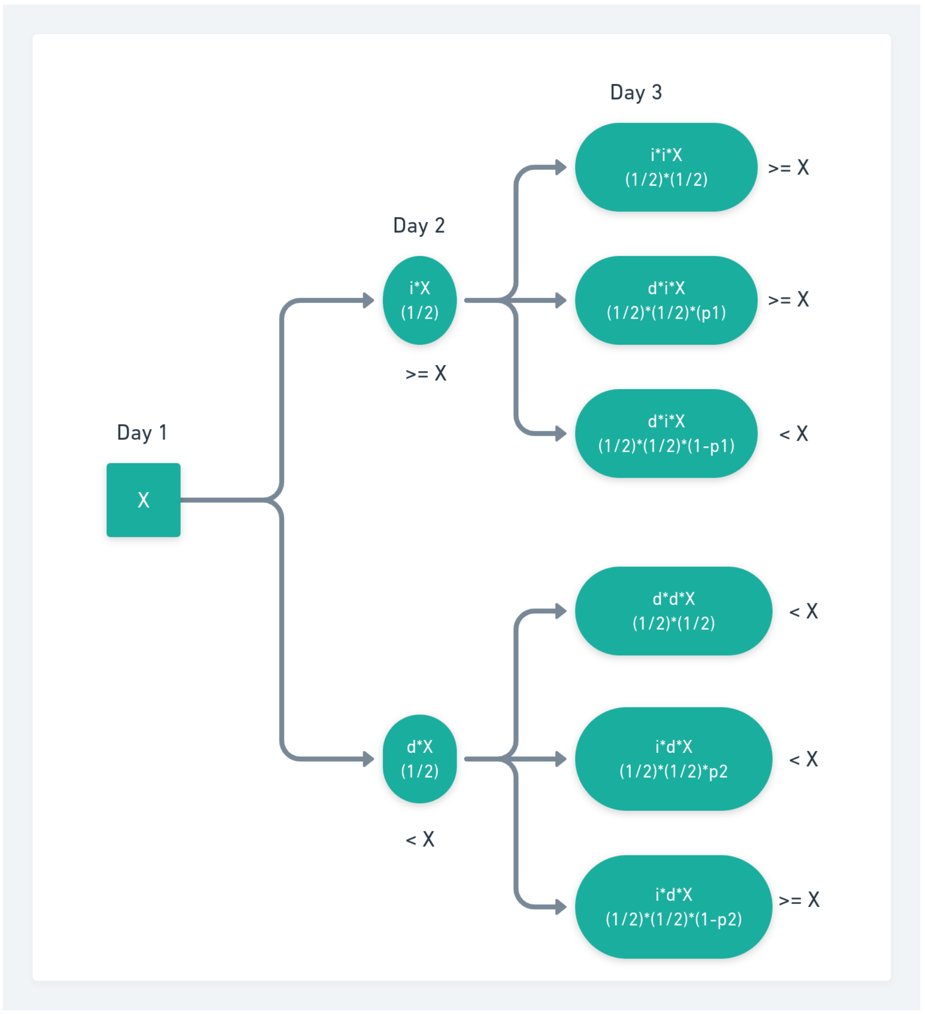


Figure 10: Different possibilities of stock price

For Day 2: There are two possibilities:- either the quoted price will be greater than X (i.e. $i * X$) or smaller than X (i.e. $d * X$) both having probability 0.5.

For Day 3:
Let the price of day 2 is denoted by Price_2 and on day 3 denoted by Price_3.

Case 1: Price_2 = $i * X$

We have three possibilities for Price_3 in this case:

1. $Price\_3 = i * Price\_2$ and $Price\_3 \geq X$
   - Quoted Price = $i * i * X$
   - Probability = $0.5 * 0.5$
   - In this case, the quoted price will be maximum because X(Price on Day 1) is incremented two times, once on the day 2 and other on the day 3, which results in the maximum possible price for the stock thus ensuring maximum profit.
2. $Price\_3 = d * Price\_2$ and $Price\_3 \geq X$
   - Quoted Price = $d * i * X$
   - Probability = $0.5 * 0.5 * p_1$
   - In this case, the price is decremented w.r.t to Day 2 but if $d * i > 1$ with probability say, p_1 $(0 < p_1 < 1)$, then the resulting quoted price on Day 3(i.e. Price_3) will be greater than the price on Day 1(i.e. X).
3. $Price\_3 = d * Price\_2$ and $Price\_3 < X$
   - Quoted Price = $d * i * X$
   - Probability = $0.5 * 0.5 * (1 - p_1)$
   - In this case, the price is decremented w.r.t to Day 2 and if $d * i < 1$, then the resulting quoted price will be less than the price on Day 1(i.e. X).

Probability(Price_3 < X) = $0.5 * 0.5 * (1 - p_1) = 0.5 * (< 0.5)$
Probability(Price_3 ≥ X) = $0.5 * 0.5 + 0.5 * 0.5 * (p_1) = 0.5 * (> 0.5)$

Thus, probability of having Price_3 ≥ X is more than the probability of having Price_3 < X. Moreover, we have maximum profit in (1), that's why we should not accept the quoted price on Day 2 and take a chance for more profit on Day 3.

Case 2: Price_2 = $d * X$

We have three possibilities for Price_3 in this case:

1. $Price\_3 = d * Price\_2$ and $Price\_3 < X$

- Quoted Price $= d * d * X$
- Probability $= 0.5 * 0.5$
- In this case, the quoted price will be minimum because X(Price on Day 1) is decremented two times, once on the day 2 and other on the day 3, which results in the minimum possible price for the stock thus leading to maximum loss.

2. $Price\_3 = i * Price\_2$ and $Price\_3 \leq X$
   - Quoted Price $= i * d * X$
   - Probability $= 0.5 * 0.5 * p_2$
   - In this case, the price is incremented w.r.t to Day 2 but if $i * d < 1$ with probability say, p_2 $(0 < p_2 < 1)$, then the resulting quoted price on Day 3(i.e. Price_3) will be less than the price on Day 1(i.e. X).

3. $Price\_3 = d * Price\_2$ and $Price\_3 > X$
   - Quoted Price $= i * d * X$
   - Probability $= 0.5 * 0.5 * (1 - p_2)$
   - In this case, the price is incremented w.r.t to Day 2 and if $i * d > 1$, then the resulting quoted price will be more than the price on Day 1(i.e. X).

Probability(Price_3 $\leq$ X) $= 0.5 * 0.5 + 0.5 * 0.5 * p_2 = 0.5 * (> 0.5)$
Probability(Price_3 $>$ X) $= 0.5 * 0.5 * (1 - p_2) = 0.5 * (< 0.5)$

Thus, probability of having Price_3 $\leq$ X is more than the probability of having Price_3 $>$ X. Moreover, we have minimum profit in (1), that's why we should accept the quoted price on Day 2.

Thus, **Optimal Strategy**:

- ***Action Profiles***: Accept the Quoted Price, Reject the Quoted Price
- ***Preferences for Maximum Profit***:
  - Reject the Quoted Price on Day 1.
  - If the Quoted Price on Day 2 is less than the X, accept the Quoted Price.
  - While if the Quoted Price on Day 2 is more than X, reject the price on Day 2 and take a chance for Day 3 for more profit, hence, accept the Quoted Price on Day 3 in this case.

1.5.2 Part B

Probability to sell the stock at maximum price possible:

As mentioned above, for this case, the ideal condition would be:

- Let the Quoted Price on Day 1 be X.
- Quoted Price on Day 2 = $i * X$ (Increment by a factor of i compared to Day 1) (Probability = 0.5)
- Quoted Price on Day 3 = $i * i * X$ (Increment by factor of i compared to Day 2) (Probability = $0.5 * 0.5$)

Hence, **the probability to sell the stock at maximum price possible will be** $0.5 * 0.5 = 0.25$

1.6 Question 6

1.6.1 Part A

Given: The quotes are always in multiples of 10, and always range in [110, 160] with equal probability, i.e., $\frac{1}{6}$th chance of drawing one of 110, 120, 130, 140, 150, 160.

Expected Payoffs

- If we have **one day remaining** to make the decision:

$$\begin{aligned} Expected\ Payoff\ (E(1)) &= \frac{1}{6} * (110 + 120 + 130 + 140 + 150 + 160) \\ &= 135 \end{aligned} \tag{1}$$

This means that we expect the **quote for that one day to be around** $135**.**

- If we have **two days remaining** to make the decision:

$$\begin{aligned} Expected\ Payoff\ (E(2)) &= \frac{1}{6} * (Quotes\ More\ Than\ E(1)) \\ &\quad + \frac{1}{6} * (No.\ of\ Quotes\ less\ than\ E(1)) * E(1) \\ &= \frac{1}{6} * (140 + 150 + 160) + \frac{1}{2} * 135 \\ &= 142.5 \end{aligned} \tag{2}$$

This means that we expect the **quote for two days to be around** $142.5**.**

Thus, **Optimal Strategy-**

We have three days to sell the stock on one of the price quoted by the buyer.

Day 1: If the Quoted Price on Day 1 will be more than E(2) i.e. 142.5 {as calculated in equation (2)}, then we accept the quote, otherwise, reject it. This means that if the quoted price is 150 or 160, then we accept the quote while if the quoted price is one of $110, 120, 130, 140$, we reject it.

Reason: The expected value of quote for the remaining two days is 142.5.

Day 2: If the Quoted Price on Day 2 will be more than E(1) i.e. 135, then we accept the quote, otherwise, reject it. This means that if the quoted price is one of $140, 150, 160$, then we accept the quote while if the quoted price is one of $110, 120, 130$, we reject it.

Reason: The expected value of quote for the remaining one day is 135.

Day 3: If rejected on the previous two days, we will have to accept the quoted price on Day 3.

1.6.2 Part B

Expected Return Value ($E(3)$) for three days can be calculated as follows:

$$\begin{aligned}
&E(3)\\
&= \frac{1}{6} * (Quotes\ more\ than\ E(2)) + \frac{1}{6} * (No.\ of\ Quotes\ less\ than\ E(2)) * E(2)\\
&= \frac{1}{6} * (150 + 160) + \frac{4}{6} * (142.5)\\
&= 146.66
\end{aligned} \tag{3}$$

So, the expected return value for the case stated in Q6: a. is 146.66. **This means that we expect the quoted price to be around 146.66, nearly 150.**

1.7 Question 7

1.7.1 Part A

Cointegration

Objective: To select a pair of stocks that are in tandem with the given scenario.

Cointegration is a statistical measure of two or more time-series variables which indicates if a linear combination of the variables is stationary. Here, '*stationary*' means that the mean and variance of the equation remains constant with time.

In the given case, the two-time series variables are log of stock prices of Bank A and Bank B.

$$Spread = log(a) - nlog(b)$$

Here,
a = Stock Price of Bank A
b = Stock Price of Bank B
n = Hedge Ratio = Constant (Assumption)

Using **regression of stock prices**, we calculate the hedge ratio so that the spread is close to zero. The residuals indicates how much the actual value of spread is **deviated**. On analysing these residuals, if we do not find a trend in the deviation, it means that the spread is deviated around zero randomly and is stationary.

Note: If two stocks are cointegrated, it means that the equation is stationary.

Now,
To check if cointegration exists, we can perform the following two tests:

- **Engle Granger Test**
- **Johansen Test**

The EG Test for cointegration identifies the **long term equilibrium relationship** between two stocks (here, stock A stock B) and tells if the relationship between them is true or not. The relationship is estimated by regressing one price of the stock on another. As stated before, the residual series contains relevant and quite significant information about the interdependent change in prices of the pair of stocks.
The **Vector Error Correction Model** can also be employed to model the stationary residual series to analyse the system reverting to the mean after a

short-term deviation.

Hence, ***cointegration*** should be used to find the most suitable pair of stocks.

1.7.2 Part B

Scenario: There are two stocks with similar impact on them by any change in the business environment.

- Stock of Bank A: Stock **A**
- Stock of Bank B: Stock **B**

In the ideal world, if the stock price of Bank A rises by $x\%$, then stock price of Bank B is also expected to rise, at least by $y\%$ and vice versa. Here, The stocks are considered **strongly correlated**.
In case of any change in the internal environment of a particular bank, the stock price of the respective bank might **deviate** from their correlation in the **short term**.

Strategy: A trading strategy popularly known as **Pairs Trading** that involves matching a long position with a short position in two stocks with high correlation.

The idea is to trade the two stocks by analysing its historical relationship of change in stock prices and during breakdown in the correlation, we bet that the gap would come back to its original state that is known as mean reversion.

Assumption: Market Neutrality i.e. the direction of market does not affect profit or loss

According to this strategy, we should match the *long position for underperforming stock* and *short position for outperforming stock.* We can also write a *call* for a security that is outperforming its pair, and match the position by writing a *put* for the pair i.e. the underpreforming stock. Subsequently, the stocks **converge to correlation** near to one and we make profit from a long position and closed short position.

How to mitigate risk?

The following cases are possible while considering the deviation of the Stock Prices:

- **If both the Stocks move in the predicted way-**

  In this case, even if the our assumption of neutrality of the market is violated slightly, we can still profit from the strategy.

- **If the underperforming stock continues to underperform-**

  In this case, until the outperforming stock falls in price, the short position/ writing a put will overtake the long position/ writing a call and can yield a profit.

- **If the outperforming stock continues to outperform-**

  In this case, until the underperforming stock rises more than rise in the outperforming stock, the long position/ writing a call will overtake the the short position/ writing a put and can yield a profit.

2 Case Study B: The Automaton

2.1 Question 1

Generation Rules:

Parent Configuration	Child value
000 (= 0)	1
001 (= 1)	1
010 (= 2)	0
011 (= 3)	0
100 (= 4)	1
101 (= 5)	0
110 (= 6)	1
111 (= 7)	0

State of the cells in the grid:

$$\text{Generation 0: } \boxed{1}\boxed{1}\boxed{0}\boxed{0}$$

$$\downarrow$$

$$\text{Generation 1: } \boxed{0}\boxed{1}\boxed{1}\boxed{1}$$

$$\downarrow$$

$$\text{Generation 2: } \boxed{0}\boxed{0}\boxed{0}\boxed{1}$$

$$\downarrow$$

$$\text{Generation 3: } \boxed{1}\boxed{1}\boxed{1}\boxed{0}$$

$$\downarrow$$

$$\text{Generation 4: } \boxed{0}\boxed{0}\boxed{1}\boxed{0}$$

2.1.1 Part A

The state of the cell in the grid after 4 generations is:

0	0	1	0

2.1.2 Part B

(1)
Replacing 1 with '|' and 0 with '.' in the cells of the grid in Generation 1 to Generation 4 we get following pattern:

```
.|||
...|
|||.
..|.
```

(2)
Code File: *Palak_Kothari_B_Q1_B_MAIN.cpp* is submitted along with report.

Description of the code file:-

```
//Data Structures

map<int,int> rule // Stores the rule table

vector<int> seed_grid // Stores the generation 0 grid

//Functions
void initialise_seed_grid(int size, vector<int> &v) // Initialises the
    seed grid

void initialise_rule(vector<int> &v) // Initialises the rule table

void next_gen(vector<int> &gen) // Calculate the next generation to
    generation stored in vector<int> gen and store it in the same vector
    gen

int main() // main() function to run the program and call other
    functions.
```

Instructions to run the program(Using Command Line):

- To compile, type command: *g++ Palak_Kothari_B_Q1_B_MAIN.cpp*
- To execute: *./a.out*

Output Of The Code File:

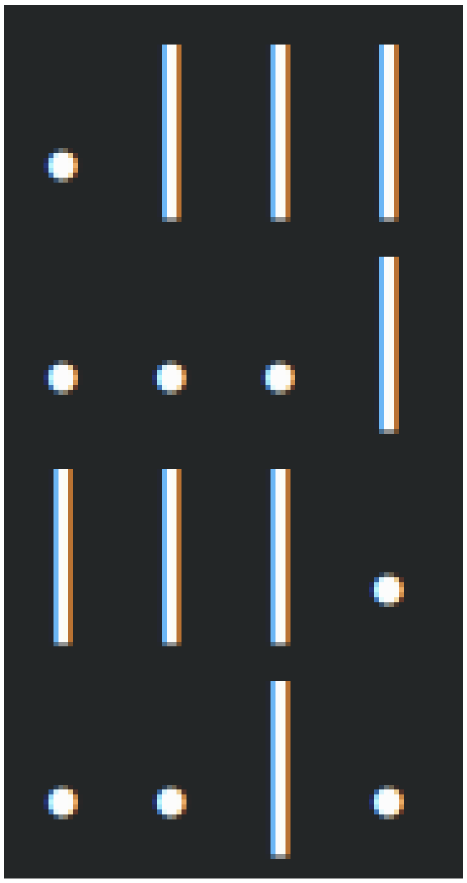


Figure 11: Output Of The Program

2.2 Question 2

Mathematics Involved in Program:
Let n be the grid size. As each cell of the grid can contain two possible values(0,1), thus total possible configuration of the grid is 2^n.

Thus, the cellular finite automata contains 2^n states, let denote it by S, each state is a combination of 0 or 1, i.e. each state represents a binary number, thus, thus each state can be represented using decimal equivalent of the binary number formed by the configuration of the grid.

PCA(Probabilistic Cellular Automata) can be represented using Markov Chains. **Say, M is the markov chain representing the PCA given in the question.**

Terminologies:

- p_{ij} is probability of transition from state i to j.
- ${p_{ij}}^{(m)}$ is probability of being in state j at time m, with starting state as i.
- $\mathbf{P}$ denotes the transition matrix for the Markov Chain.
- $$ {p_{ij}}^{(m)} \ is\ i, j_{th}\ entry\ of\ \mathbf{P}^{(m)} \tag{4}$$
- ${\pi_j}^{(m)}$ is probability of being in state j at time m.
  $\implies {\pi_j}^{(m)} = Pr\{X_m = j\} = \sum_i {\pi_j}^{(m-1)} * p_{ij}$
  $\implies {\pi_j}^{(m)} = \pi^{(0)}\mathbf{P}^m$
- Let k be the starting state of the automata, then, $\pi^{(0)} = \mathbf{e}_k$,where $\mathbf{e}_k$ is a column vector with 1 in k_{th} row and 0 in all others.

$$\begin{aligned} As,& \\ &\pi_j^{(m)} = \pi^{(0)}\mathbf{P}^m, \\ &\implies {\pi_j}^{(m)} = {p_{kj}}^{(m)} \end{aligned} \tag{5}$$

Properties of PCA's Markov Chain:

- **Irreducible:** As p_{ij} is greater than 0 $\forall\ i,\ j$, for the given rules table. Thus the markov chain M is irreducible.
- **Positive Recurrent:** As Markov Chain M is irreducible and finite $\implies$ Markov Chain M is Positive Recurrent.
- **Aperiodic:** As $\mathbf{P} > \mathbf{O}$ ($\mathbf{P}$ is the Transition Matrix) $\implies \forall$ pow ≥ 1, $\mathbf{p}^{pow} > \mathbf{O}$, where $\mathbf{O}$ is Zero Matrix. Thus, Markov Chain M is Aperiodic.

As Markov Chain M is irreducible, positive recurrent and aperiodic $\implies$ **limiting distribution exists.**

About Program:

1. We need to calculate matrix $\mathbf{P}$, takes $O(n * 2^{2*n})$ time. As their are total 2^n states, thus total 2^{2*n} pairs and for each pair we take $O(n)$ time to calculate transition probabilities.
2. After this we need to calculate $\mathbf{P}^{(k)}$ for some k. For matrix multiplication of two matrices take $O(2^{2*n})$ time, and using binary exponentiation for calculating matrix raised to power k, takes $O(log(k) * 2^{2*n})$ time.
3. Time Complexity of the program will be = O($log(k) * 2^{2*n} + n * 2^{2*n}$)

2.2.1 Part A

Grid of size is 4, thus number of states are 16(from 0 to 15).
Initial state is: $\boxed{0}\boxed{1}\boxed{1}\boxed{0}$ i.e. = 6
$\pi^{(0)} = \mathbf{e}_6$,
Using equation (5):
${\pi_j}^{(m)} = {p_{6j}}^{(m)}$
Here m = 1, as we need to calculate for first generation.
Code File: *Palak_Kothari_B_Q2_A_MAIN.cpp* is submitted along with report.

Description of the code file:-

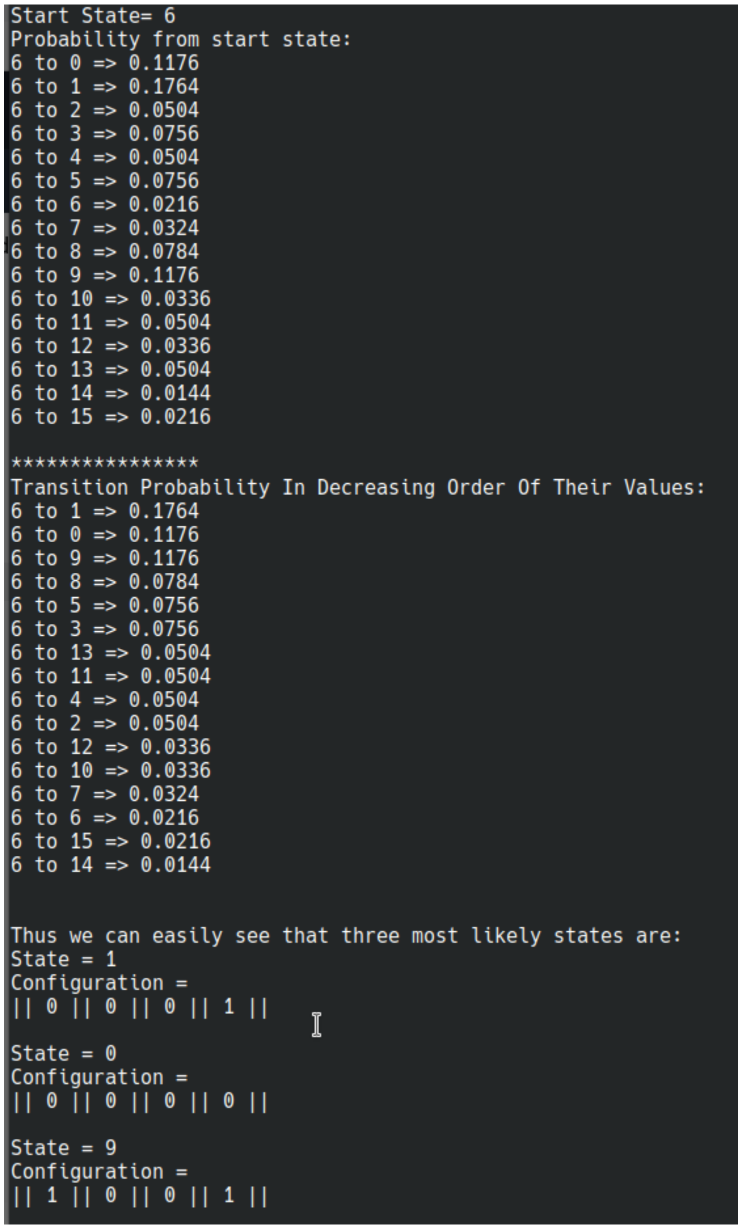
```
//Data Structures

vector<int> seed_grid; //contains the seed generation configuration of
    the grid

map<int,double> rule; //contains probability for having 0 in the cell in
    next generation

map<int, vector<int>> state;//state number to configuration

vector<vector<double>> prob; //Stores the transition matrix

//Main Functions Of The Program

//To initialise the seed grid
void initialise_seed_grid(int size, vector<int> &v)

//To Fill The Rule Table
void initialise_rule(vector<double> &v)

//Calculates Transition Matrix
void preprocess_probability(int n)

//Multipluy two matrices and store them 2-D vector prob_k
void multiply(vector<vector<double>> &res, vector<vector<double>> a,
    vector<vector<double>> b)

//Calculate M^k in O(logk) time where M is a matrix
void binary_exponentiation(int n, int st, int k)

int main() // main() function to run the program and call other
    functions.
```

Instructions to run the program(Using Command Line):

- To compile, type command: *g++ Palak_Kothari_B_Q2_A_MAIN.cpp*

- To execute: *./a.out*

Output Of The Code File:



```
Start State= 6
Probability from start state:
6 to 0 => 0.1176
6 to 1 => 0.1764
6 to 2 => 0.0504
6 to 3 => 0.0756
6 to 4 => 0.0504
6 to 5 => 0.0756
6 to 6 => 0.0216
6 to 7 => 0.0324
6 to 8 => 0.0784
6 to 9 => 0.1176
6 to 10 => 0.0336
6 to 11 => 0.0504
6 to 12 => 0.0336
6 to 13 => 0.0504
6 to 14 => 0.0144
6 to 15 => 0.0216

****************
Transition Probability In Decreasing Order Of Their Values:
6 to 1 => 0.1764
6 to 0 => 0.1176
6 to 9 => 0.1176
6 to 8 => 0.0784
6 to 5 => 0.0756
6 to 3 => 0.0756
6 to 13 => 0.0504
6 to 11 => 0.0504
6 to 4 => 0.0504
6 to 2 => 0.0504
6 to 12 => 0.0336
6 to 10 => 0.0336
6 to 7 => 0.0324
6 to 6 => 0.0216
6 to 15 => 0.0216
6 to 14 => 0.0144


Thus we can easily see that three most likely states are:
State = 1
Configuration =
|| 0 || 0 || 0 || 1 ||

State = 0
Configuration =
|| 0 || 0 || 0 || 0 ||

State = 9
Configuration =
|| 1 || 0 || 0 || 1 ||
```

Figure 12: Output Of The Program

Three Most Likely Configurations Are:

1. State 1, with probability = 0.1764: $\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 1 \\ \hline\end{array}$
2. State 0, with probability = 0.1176: $\begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline\end{array}$
3. State 9, with probability = 0.1176: $\begin{array}{|c|c|c|c|}\hline 1 & 0 & 0 & 1 \\ \hline\end{array}$

2.2.2 Part B

Grid of size is 8, thus number of states are 256(from 0 to 255).
Initial state is: $\boxed{1 \mid 0 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0}$ i.e. = 152
$\pi^{(0)} = \mathbf{e}_{152}$,
Using equation (5):
${\pi_j}^{(m)} = {p_{152j}}^{(m)}$

1. **For k = 5**, set m = 5
2. **For k = 15**, set m = 15
3. **For k** = inf, As proved above Markov Chain M of PCA has limiting distribution. Algorithm to find limiting distribution:
   (a) Set m to some positive integer, say val.
   (b) Find Transition Matrix, say it is P_{val}
   (c) Now set m to $val + 1$, find Transition Matrix, say it is P_{val+1}
   (d) Now if $P_{val} == P_{val+1}$, then P_{val} represents the limiting distribution
   (e) Otherwise, set val to $2 * val$.

   This algorithm is bound to terminate as in our Markov Chain, M, limiting distribution exists, i.e. $P^{(val)} == P^{(val+1)}$, $\forall val \geq num$ for some positive integer num.

Code File: *Palak_Kothari_B_Q2_B_MAIN.cpp* is submitted along with report.

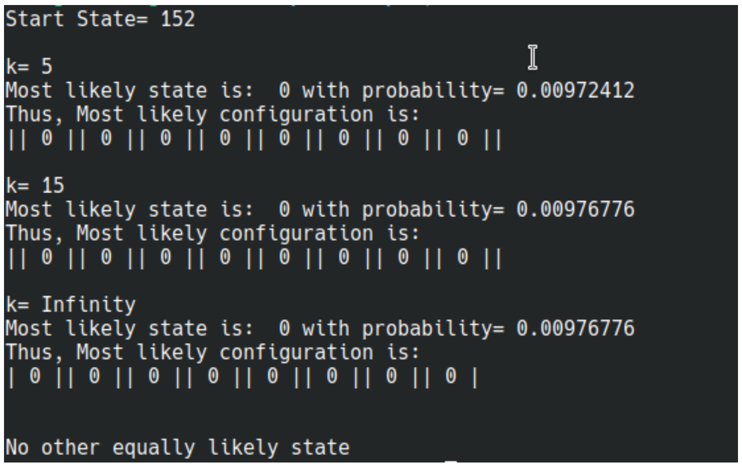
Description of the code file:-

```
//To Calculate Limiting Distribution, case K = infinity
void limiting_distribution(vector<pair<double,int>> &pi, int n, int st)

//Other Functions and Datastructures are same as PS2_Q2A.cpp
```

Instructions to run the program(Using Command Line):

- To compile, type command: *g++ Palak_Kothari_B_Q2_B_MAIN.cpp*
- To execute: *./a.out*

Output Of The Code File:



```
Start State= 152

k= 5
Most likely state is:  0 with probability= 0.00972412
Thus, Most likely configuration is:
|| 0 || 0 || 0 || 0 || 0 || 0 || 0 || 0 ||

k= 15
Most likely state is:  0 with probability= 0.00976776
Thus, Most likely configuration is:
|| 0 || 0 || 0 || 0 || 0 || 0 || 0 || 0 ||

k= Infinity
Most likely state is:  0 with probability= 0.00976776
Thus, Most likely configuration is:
| 0 || 0 || 0 || 0 || 0 || 0 || 0 || 0 |


No other equally likely state
```

Figure 13: Output Of The Program

Most Likely Configuration Of The Grid:

1. **For k = 5**,

   State 0, with probability = 0.00972412:

0	0	0	0	0	0	0	0

2. **For k = 15**,

   State 0, with probability = 0.00976776:

0	0	0	0	0	0	0	0

3. **For k =** inf

   State 0, with probability = 0.00976776:

0	0	0	0	0	0	0	0