In [1]:
```
# Import Necessary Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
import operator as op
from functools import reduce
from decimal import Decimal
```

In [2]:
```
#Inputs

x0_val = 95 #Current Stock Price
x0_val_ans4 = 105 #Assumed Stock Price for Answer 4
k_val = 105 #Strike Price
t_val1 = 3 #Steps in the Pricing Process (No. of Steps per Year x Years to Maturity) (Ans 1 Part a)
t_val2 = 2 #Steps in the Pricing Process (No. of Steps per Year x Years to Maturity) (Ans 2 Part a)
r_val = 0 #Risk-free Rate per Price Step (Calculated as Risk-free Rate per year/No. of Steps per year)
r_val_ans4 = 0.01 #Assumed Risk-free Rate per Price Step for Answer 4
u_val = 1.14 #Up Movement per Step in Binomial Model
```

In [3]:
```
#Combinatorics Function

def nCr(n, r):
    r = min(r, n-r)
    numer = reduce(op.mul, range(n, n-r, -1), 1)
    denom = reduce(op.mul, range(1, r+1), 1)
    return (numer // denom)
```

In [4]:
```
#Asset Price Binomial Tree Function:

def Asset_Px_Binomial_Tree(t, u, d, x0):
    tree_holder = np.zeros([t+1, t+1])
    freq = np.zeros(t+1)
    d = 1/u

    for i in range(t+1):
        for j in range(i+1):
            tree_holder[j,i]=x0*(d**j)*(u**(i-j))
        freq[i]= Decimal(nCr(t, i))

    tree_holder = pd.DataFrame(tree_holder.round(2))
    return (tree_holder, freq)
```

In [5]:
```
#Dataframe of Terminal Asset Prices from Binomial Tree:

def Terminal_Px_Extractor(binomial_tree):
    terminal_px = pd.DataFrame(np.vstack([binomial_tree.iloc[:,-1].values]).T,
                               columns = ['X(w)'])
    return terminal_px
```

In [6]:
```
#European Option Price Function

def European_Option_Price(x0, k, u, t, r, option_type):

    d = 1/u
    p = (1 - d)/(u - d)

    if option_type == "call":
        price = reduce(lambda a, y: a + max(x0*(u**y)*(d**(t-y)) - k, 0) * nCr(t, y)*(p**y)*((1-p)**(t-y))
                       * (1/((1+r)**t)), [0]+list(range(t+1)))
        return price

    elif option_type == "put":
        price = reduce(lambda a, y: a + max(k - x0*(u**y)*(d**(t-y)), 0) * nCr(t, y)*(p**y)*((1-p)**(t-y))
                       * (1/((1+r)**t)), [0]+list(range(t+1)))
        return price

    else:
        price = print("Input format error")
        return price
```

In [7]:
```
#European Option Value Tree Function:

def European_Optionval_Tree(stockpx_binomial_tree, k, u, t, r, option_type):

    european_optionval_tree = pd.DataFrame()

    if option_type == "call":
        for i in range(stockpx_binomial_tree.shape[0]):
            for j in range(stockpx_binomial_tree.shape[1]):
                if i <= j:
                    european_optionval_tree.at[i, j] = European_Option_Price(stockpx_binomial_tree.at[i, j],
                                                                           k, u, (t-j), r, option_type)
                else:
                    european_optionval_tree.at[i, j] = 0
        return european_optionval_tree.round(3)

    elif option_type == "put":
        for i in range(stockpx_binomial_tree.shape[0]):
            for j in range(stockpx_binomial_tree.shape[1]):
                if i <= j:
                    european_optionval_tree.at[i, j] = European_Option_Price(stockpx_binomial_tree.at[i, j],
                                                                           k, u, (t-j), r, option_type)
                else:
                    european_optionval_tree.at[i, j] = 0
        return european_optionval_tree.round(3)

    else:
        european_optionval_tree = print("Input format error")
        return european_optionval_tree.round(3)
```

In [8]:
```
#Code Test - Verifying Values Using Example on Pages 6-7 of M5 Notes

call_m5_eg = European_Option_Price(100, 110, 1.2, 2, 0, "call")
put_m5_eg = European_Option_Price(100, 110, 1.2, 2, 0, "put")

call_m5_eg, put_m5_eg
```

Out[8]:
```
(7.024793388429751, 17.024793388429746)
```

Answer 1 European Call Option (N = 3)

Part (a) European Call Option Price

In [9]:
```
call_price = European_Option_Price(x0_val, k_val, u_val, t_val1, r_val, "call")

print("The price of the European call option with the given parameters is", "{:.3f}".format(call_price))
```

```
The price of the European call option with the given parameters is 4.799
```

Part (b) Call Option Value H(w) for Each Price Path

In [10]:
```
#Stock Price Evolution

stockpx_binomial_tree1, freq_stockpx1 = Asset_Px_Binomial_Tree(t_val1, u_val, (1/u_val), x0_val)
stockpx_binomial_tree1
```

Out[10]:

	0	1	2	3
0	95.0	108.30	123.46	140.75
1	0.0	83.33	95.00	108.30
2	0.0	0.00	73.10	83.33
3	0.0	0.00	0.00	64.12

In [11]:
```
#Pay-off of the European Call Option for Each Price Path

call_payoffs = Terminal_Px_Extractor(stockpx_binomial_tree1)
call_payoffs['H(w)'] = [max(i - k_val, 0) for i in call_payoffs['X(w)']]

print('H(w) (call) = Max(Terminal Price - Strike, 0)')
call_payoffs
```

```
H(w) (call) = Max(Terminal Price - Strike, 0)
```

Out[11]:

	X(w)	H(w)
0	140.75	35.75
1	108.30	3.30
2	83.33	0.00
3	64.12	0.00

In [12]:
```
#Value of the European Call Option at Each Node
call_optionval_tree = European_Optionval_Tree(stockpx_binomial_tree1, k_val, u_val, t_val1, r_val, "call")
call_optionval_tree

#def myfunc(x):
#    c1=np.triu(np.ones(call_optionval_tree.shape),1).astype(np.bool)
#    c2=np.tril(np.ones(call_optionval_tree.shape),-1).astype(np.bool)
#    col1='color:black'
#    col2='color:white'
#    col3='color:black'
#    df1 = pd.DataFrame(np.select([c1,c2],[col1,col2],col3),columns=x.columns,index=x.index)
#    return df1

#call_optionval_tree.style.apply(myfunc,axis=None)
```

Out[12]:

	0	1	2	3
0	4.799	9.449	18.460	35.75
1	0.000	0.720	1.542	3.30
2	0.000	0.000	0.000	0.00
3	0.000	0.000	0.000	0.00

Answer 2 European Put Option (N = 2)

Part (a) European Put Option Price

In [13]:
```
put_price = European_Option_Price(x0_val, k_val, u_val, t_val2, r_val, "put")

print("The price of the European put option with the given parameters is", "{:.3f}".format(put_price))
```

```
The price of the European put option with the given parameters is 14.031
```

Part (b) Stock Price Evolution, Option Pay-off, Option Value, Hedging Strategy

In [14]:
```
#Stock Price Evolution

stockpx_binomial_tree2, freq_stockpx2 = Asset_Px_Binomial_Tree(t_val2, u_val, (1/u_val), x0_val)
stockpx_binomial_tree2
```

Out[14]:

	0	1	2
0	95.0	108.30	123.46
1	0.0	83.33	95.00
2	0.0	0.00	73.10

In [15]:
```
#Pay-off of the European Put Option for Each Price Path

put_payoffs = Terminal_Px_Extractor(stockpx_binomial_tree2)
put_payoffs['H(w)'] = [max(k_val - i, 0) for i in put_payoffs['X(w)']]

print('H(w) (put) = Max(Strike - Terminal Price, 0)')
put_payoffs
```

```
H(w) (put) = Max(Strike - Terminal Price, 0)
```

Out[15]:

	X(w)	H(w)
0	123.46	0.0
1	95.00	10.0
2	73.10	31.9

In [16]:
```
#Value of the European Put Option at Each Node

put_optionval_tree = European_Optionval_Tree(stockpx_binomial_tree2, k_val, u_val, t_val2, r_val, "put")
put_optionval_tree
```

Out[16]:

	0	1	2
0	14.031	5.327	0.0
1	0.000	21.670	10.0
2	0.000	0.000	31.9

In [17]:
```
#Table of Stock Price Xt(w) Evolution and Option Pay-off

stockpx_table = pd.DataFrame(columns = ['w', 'X0(w)', 'X1(w)', 'X2(w)', 'H(w) = Max(K - X2(w), 0)'])

stockpx_table.at[0, 'w'] = '(u,u)'
stockpx_table.at[1, 'w'] = '(u,d)'
stockpx_table.at[2, 'w'] = '(d,u)'
stockpx_table.at[3, 'w'] = '(d,d)'

#Stock Price at t = 0
for i in range(4):
    stockpx_table.at[i, 'X0(w)'] = stockpx_binomial_tree2.at[0,0]

#Stock Price at t = 1
for i in range(2):
    stockpx_table.at[i, 'X1(w)'] = stockpx_binomial_tree2.at[0,1]

for i in range(2,4):
    stockpx_table.at[i, 'X1(w)'] = stockpx_binomial_tree2.at[1,1]

#Stock Price at t = 2
stockpx_table.at[0, 'X2(w)'] = stockpx_binomial_tree2.at[0,2]
stockpx_table.at[1, 'X2(w)'] = stockpx_binomial_tree2.at[1,2]
stockpx_table.at[2, 'X2(w)'] = stockpx_binomial_tree2.at[1,2]
stockpx_table.at[3, 'X2(w)'] = stockpx_binomial_tree2.at[2,2]

#Put Option Pay-off H(w)
for i in range(4):
    stockpx_table.at[i, 'H(w) = Max(K - X2(w), 0)'] = max(k_val - stockpx_table.at[i, 'X2(w)'], 0)

#stockpx_table.index += 1
stockpx_table
```

Out[17]:

	w	X0(w)	X1(w)	X2(w)	H(w) = Max(K - X2(w), 0)
0	(u,u)	95	108.3	123.46	0
1	(u,d)	95	108.3	95	10
2	(d,u)	95	83.33	95	10
3	(d,d)	95	83.33	73.1	31.9

In [18]:
```
#Table of Put Option Value Vt(w) Evolution

putval_table = pd.DataFrame(columns = ['w', 'V0(w)', 'V1(w)', 'V2(w)'])

putval_table.at[0, 'w'] = '(u,u)'
putval_table.at[1, 'w'] = '(u,d)'
putval_table.at[2, 'w'] = '(d,u)'
putval_table.at[3, 'w'] = '(d,d)'

#Put Option Price at t = 0
for i in range(4):
    putval_table.at[i, 'V0(w)'] = put_optionval_tree.at[0,0].round(3)

#Put Option Price at t = 1
for i in range(2):
    putval_table.at[i, 'V1(w)'] = put_optionval_tree.at[0,1].round(3)

for i in range(2,4):
    putval_table.at[i, 'V1(w)'] = put_optionval_tree.at[1,1].round(3)

#Put Option Price at t = 2
putval_table.at[0, 'V2(w)'] = put_optionval_tree.at[0,2].round(3)
putval_table.at[1, 'V2(w)'] = put_optionval_tree.at[1,2].round(3)
putval_table.at[2, 'V2(w)'] = put_optionval_tree.at[1,2].round(3)
putval_table.at[3, 'V2(w)'] = put_optionval_tree.at[2,2].round(3)

putval_table
```

Out[18]:

	w	V0(w)	V1(w)	V2(w)
0	(u,u)	14.031	5.327	0
1	(u,d)	14.031	5.327	10
2	(d,u)	14.031	21.67	10
3	(d,d)	14.031	21.67	31.9

In [19]:
```
#Hedging Strategy at Each Node

hedging_strat_table = pd.DataFrame(columns = ['w', 'Q1(w)', 'Q2(w)'])

hedging_strat_table.at[0, 'w'] = '(u,u)'
hedging_strat_table.at[1, 'w'] = '(u,d)'
hedging_strat_table.at[2, 'w'] = '(d,u)'
hedging_strat_table.at[3, 'w'] = '(d,d)'

for j in range(1, hedging_strat_table.shape[1]):
    for i in range(hedging_strat_table.shape[0]):
        hedging_strat_table.iloc[i, j] = ((putval_table.iloc[i, j+1] - putval_table.iloc[i, j])/
                                          (stockpx_table.iloc[i, j+1] - stockpx_table.iloc[i, j])).round(3)

hedging_strat_table
```

Out[19]:

	w	Q1(w)	Q2(w)
0	(u,u)	-0.654	-0.351
1	(u,d)	-0.654	-0.351
2	(d,u)	-0.655	-1
3	(d,d)	-0.655	-1

Answer 3 Market Completeness Analysis

Part (a) (2x2) A Matrix Construction

In [20]:
```
A = pd.DataFrame(columns = ['State','Xt(w)', 'B'])
A['State'] = ['u', 'd']
A.set_index('State', inplace = True)
A = A.fillna('-')
A
```

Out[20]:

State	Xt(w)	B
u	-	-
d	-	-

Part (b) Populating Matrix A

In [21]:
```
#We take the node X1({(d,u), (d,d)})

node = stockpx_binomial_tree2.at[1, 1]

for i in range(A.shape[0]):
    A['Xt(w)'].iloc[i] = stockpx_binomial_tree2.at[i+1, 2]

for i in range(A.shape[0]):
    A['B'] = 100

A
```

Out[21]:

State	Xt(w)	B
u	95	100
d	73.1	100

Part (c) Constructing and Populating the b Matrix

In [22]:
```
b = pd.DataFrame(columns = ['State', 'Vt(w)'])
b['State'] = ['u', 'd']
b.set_index('State', inplace = True)

for i in range(b.shape[0]):
    b.iloc[i] = put_optionval_tree.at[i+1, 2]

b
```

Out[22]:

State	Vt(w)
u	10
d	31.9

Part (e) Solving for x

In [23]:
```
#Solving the Equation Ax = b for x

from numpy import linalg
linalg.solve(A.to_numpy(dtype = 'float'), b.to_numpy(dtype = 'float'))
```

Out[23]:
```
array([[-1.  ],
       [ 1.05]])
```

Answer 4 Put-Call Parity

Part (a) Comparing Portfolio with Long Call and Short Put to Portfolio with Long Stock and Short Bond

In [24]:
```
#Stock Price Binomial Tree with X0 = 105

stockpx_binomial_tree3, freq_stockpx3 = Asset_Px_Binomial_Tree(t_val2, u_val, 1/u_val, x0_val_ans4)
stockpx_binomial_tree3
```

Out[24]:

	0	1	2
0	105.0	119.70	136.46
1	0.0	92.11	105.00
2	0.0	0.00	80.79

In [25]:
```
#European Call Option Value Binomial Tree

call_optionval_tree2 = European_Optionval_Tree(stockpx_binomial_tree3, k_val, u_val,
                                               t_val2, r_val_ans4, "call")
call_optionval_tree2
```

Out[25]:

	0	1	2
0	6.734	14.554	31.46
1	0.000	0.002	0.00
2	0.000	0.000	0.00

In [26]:
```
#European Put Option Value Binomial Tree

put_optionval_tree2 = European_Optionval_Tree(stockpx_binomial_tree3, k_val, u_val,
                                              t_val2, r_val_ans4, "put")
put_optionval_tree2
```

Out[26]:

	0	1	2
0	6.734	0.000	0.00
1	0.000	12.765	0.00
2	0.000	0.000	24.21

Part (a-i.) Value of Portfolio with 1 Long Call and 1 Short Put

In [27]:

```
#Option Portfolio Value at Each Node

option_portfolio_val = call_optionval_tree2 - put_optionval_tree2
option_portfolio_val
```

Out[27]:

	0	1	2
0	0.0	14.554	31.46
1	0.0	-12.763	0.00
2	0.0	0.000	-24.21

Part (a-ii.) Value of Portfolio with 1 Long Stock and K dollars Borrowing (Short Bond)

In [28]:

```
#Stock-Bond Portfolio Value at Each Node

stock_bond_portfolio_val = pd.DataFrame()

for j in range(stockpx_binomial_tree3.shape[1]):
    for i in range(stockpx_binomial_tree3.shape[0]):

        if i <=j:
            stock_bond_portfolio_val.at[i,j] = (stockpx_binomial_tree3.at[i,j]
            - k_val/((1+r_val_ans4)**(t_val2-j))).round(3)

        else:
            stock_bond_portfolio_val.at[i,j] = 0

stock_bond_portfolio_val
```

Out[28]:

	0	1	2
0	2.069	15.74	31.46
1	0.000	-11.85	0.00
2	0.000	0.00	-24.21

Value of the Two Portfolios across the Three States

In [29]:

```
#Portfolio Value for the 3 Final States

portfolio_val_comparison_table = pd.DataFrame(columns = ['w', 'X2(w)', 'K' , '(C-P)2(w)', '(S-B)2(w)'])

portfolio_val_comparison_table.at[0, 'w'] = '(u,u)'
portfolio_val_comparison_table.at[1, 'w'] = '(u,d)'
portfolio_val_comparison_table.at[2, 'w'] = '(d,u)'
portfolio_val_comparison_table.at[3, 'w'] = '(d,d)'

#Terminal Stock Price
portfolio_val_comparison_table.at[0, 'X2(w)'] = stockpx_binomial_tree3.at[0,2].round(3)
portfolio_val_comparison_table.at[1, 'X2(w)'] = stockpx_binomial_tree3.at[1,2].round(3)
portfolio_val_comparison_table.at[2, 'X2(w)'] = stockpx_binomial_tree3.at[1,2].round(3)
portfolio_val_comparison_table.at[3, 'X2(w)'] = stockpx_binomial_tree3.at[2,2].round(3)

#Strike Price
for i in range(4):
    portfolio_val_comparison_table.at[i, 'K'] = k_val

#Option Portfolio Value at t = 2
portfolio_val_comparison_table.at[0, '(C-P)2(w)'] = option_portfolio_val.at[0,2].round(3)
portfolio_val_comparison_table.at[1, '(C-P)2(w)'] = option_portfolio_val.at[1,2].round(3)
portfolio_val_comparison_table.at[2, '(C-P)2(w)'] = option_portfolio_val.at[1,2].round(3)
portfolio_val_comparison_table.at[3, '(C-P)2(w)'] = option_portfolio_val.at[2,2].round(3)

#Option Portfolio Value at t = 2
portfolio_val_comparison_table.at[0, '(S-B)2(w)'] = stock_bond_portfolio_val.at[0,2].round(3)
portfolio_val_comparison_table.at[1, '(S-B)2(w)'] = stock_bond_portfolio_val.at[1,2].round(3)
portfolio_val_comparison_table.at[2, '(S-B)2(w)'] = stock_bond_portfolio_val.at[1,2].round(3)
portfolio_val_comparison_table.at[3, '(S-B)2(w)'] = stock_bond_portfolio_val.at[2,2].round(3)

portfolio_val_comparison_table
```

Out[29]:

	w	X2(w)	K	(C-P)2(w)	(S-B)2(w)
0	(u,u)	136.46	105	31.46	31.46
1	(u,d)	105	105	0	0
2	(d,u)	105	105	0	0
3	(d,d)	80.79	105	-24.21	-24.21

Part (b) Verifying Put-Call Parity for Options in Questions 1 and 2 (Taking N = 2)

In [30]:

```
#Stock Price Evolution

stockpx_binomial_tree2
```

Out[30]:

	0	1	2
0	95.0	108.30	123.46
1	0.0	83.33	95.00
2	0.0	0.00	73.10

In [31]:

```
#Value of the European Call Option at Each Node

call_optionval_tree3 = European_Optionval_Tree(stockpx_binomial_tree2, k_val, u_val, t_val2,
                                               r_val_ans4, "call")
call_optionval_tree3
```

Out[31]:

	0	1	2
0	3.952	8.542	18.46
1	0.000	0.000	0.00
2	0.000	0.000	0.00

In [32]:

```
#Value of the European Put Option at Each Node

put_optionval_tree = European_Optionval_Tree(stockpx_binomial_tree2, k_val, u_val, t_val2,
                                             r_val_ans4, "put")
put_optionval_tree
```

Out[32]:

	0	1	2
0	13.755	5.274	0.0
1	0.000	21.455	10.0
2	0.000	0.000	31.9

In [33]:

```
#Option Portfolio Value at Each Node

option_portfolio_val2 = call_optionval_tree3 - put_optionval_tree
option_portfolio_val2
```

Out[33]:

	0	1	2
0	-9.803	3.268	18.46
1	0.000	-21.455	-10.00
2	0.000	0.000	-31.90

In [34]:

```
#Stock-Bond Portfolio Value at Each Node

stock_bond_portfolio_val2 = pd.DataFrame()

for j in range(stockpx_binomial_tree2.shape[1]):
    for i in range(stockpx_binomial_tree2.shape[0]):

        if i <=j:
            stock_bond_portfolio_val2.at[i,j] = (stockpx_binomial_tree2.at[i,j]
            - k_val/((1+r_val_ans4)**(t_val2-j))).round(3)

        else:
            stock_bond_portfolio_val2.at[i,j] = 0

stock_bond_portfolio_val2
```

Out[34]:

	0	1	2
0	-7.931	4.34	18.46
1	0.000	-20.63	-10.00
2	0.000	0.00	-31.90

In [35]:

```
#Difference between Option Portfolio Value and Stock-Bond Portfolio Value

diff = option_portfolio_val2 - stock_bond_portfolio_val2
diff.round(3)
```

Out[35]:

	0	1	2
0	-1.872	-1.072	0.0
1	0.000	-0.825	0.0
2	0.000	0.000	0.0

In [ ]: