Q1) American Call Option

Code-:

```
import numpy as np
import pandas as pd
import itertools
def function(X0, K, T, r, u, d, N, type_put_call = "call"):

  print("Options Calculator:")
  print("X0 : %2d, K : %2d, u : %5.2f, d : %5.4f, N : %2d" % (X0, K,u,d,N))
  p_star = (1-d)/(u-d)
  p_up = p_star
  p_down = 1 - p_star
  print("p* : %5.4f" % (p_star))

  columns=['Value','difference']+['X'+str(i) for i in range(N+1)]+['V'+str(i) for i in range(N+1)]+['HS'+str(i) for i in
range(1,N+1)]
  table = pd.DataFrame(0.0,index= list(itertools.product("ud", repeat=N)),columns=columns)
  table.index=[''.join(i) for i in table.index]
  table['X0']=float(X0)
  print(table)
  for index,col in table.iterrows():
    base_price = X0
    for t in range (1,N+1):
      #
      i = index[0:t].count("u")
      j = t - i
      stock_price = base_price * u**i * d**j
      table.at[index,'X'+str(t)] = stock_price
      #
    if type_put_call == "call":
      difference = max(stock_price - K,0)
    if type_put_call == "put":
      difference = max(K - stock_price,0)
    table.at[index,'difference'] = difference
    #
    table.at[index,'Value'] = difference
    table.at[index,'V'+str(N)] = difference

  for t in range (N-1,-1,-1):
```

```
    for index,col in table.iterrows():
      #
      index_char = index[t]
      #
      if index_char == "u":
        change_char = "d"
        new_index = index[0:t] + change_char + index[t+1:]
        option_price = p_up * table.loc[index,'V'+str(t+1)] + table.loc[new_index,'V'+str(t+1)] * p_down
        if type_put_call == "call": option_price = max(0, option_price, table.loc[index,'X'+str(t)] - K)
        if type_put_call == "put": option_price = max(0, option_price, K - table.loc[index,'X'+str(t)])
      if index_char == "d":
        change_char = "u"
        new_index = index[0:t] + change_char + index[t+1:]
        option_price = p_up * table.loc[new_index,'V'+str(t+1)] + table.loc[index,'V'+str(t+1)] * p_down
        if type_put_call == "call":
            option_price = max(0, option_price, table.loc[index,'X'+str(t)] - K)
        if type_put_call == "put":
            option_price = max(0, option_price, K - table.loc[index,'X'+str(t)])

      table.at[index,'V'+str(t)] = option_price
      V0 = option_price

  for index,col in table.iterrows():
    for t in range (N,0,-1):
      #
      if (table.loc[index,'X'+str(t)] - table.loc[index,'X'+str(t-1)]) != 0:
        hedging_strategy = (table.loc[index,'V'+str(t)] - table.loc[index,'V'+str(t-1)]) / (table.loc[index,'X'+str(t)] -
table.loc[index,'X'+str(t-1)])

        table.at[index,'HS'+str(t)] = hedging_strategy

  return table

X0=95
K=90
T= 1 # year
r= 0  #
u=1.17 #(Group 7)
d =1/u
N = 5
```

```
# ------------------ Q1  ------------------------
function(X0, K, T, r, u, d, N, "call")


# The price of the option is:
# V0 : $16.03
```

a)
Utilized Python to model a 5-step cost structure for a 1-year American call option on a value of the stock selling at $95 with a strike price of $90 and an anticipated interest rate of 0%.
The price of the option is:
V0 : $16.03

Code in python attached

b) Is there any point time where we, as buyers of the option, benefit from early exercise?
=> Given the computed values above, there is no moment in time when the buyer can gain from buying options early because the option's value rises as it approaches age.

c) Explain whether your previous answer is (or is not) always the case.
=> In real markets, the feature of little gain from early exercise isn't usually true. When the option is fully in the money and close to expiry, its time value is insignificant, a buyer (or trader) would gain from executing the call option early. Another motivation for early exercise is that the buyer will then be able to take advantage of a potential dividend on the underlying stock if the option is exercised early.

Q2) American Put option
Code-:

```
import numpy as np
import pandas as pd
import itertools
def function(X0, K, T, r, u, d, N, type_put_call = "call"):

  print("Options Calculator:")
  print("X0 : %2d, K : %2d, u : %5.2f, d : %5.4f, N : %2d" % (X0, K,u,d,N))
  p_star = (1-d)/(u-d)
  p_up = p_star
  p_down = 1 - p_star
  print("p* : %5.4f" % (p_star))

  columns=['Value','difference']+['X'+str(i) for i in range(N+1)]+['V'+str(i) for i in range(N+1)]+['HS'+str(i) for i in
range(1,N+1)]
  table = pd.DataFrame(0.0,index= list(itertools.product("ud", repeat=N)),columns=columns)
  table.index=[''.join(i) for i in table.index]
  table['X0']=float(X0)
  print(table)
  for index,col in table.iterrows():
    base_price = X0
    for t in range (1,N+1):
      #
      i = index[0:t].count("u")
      j = t - i
```

```
    stock_price = base_price * u**i * d**j
    table.at[index,'X'+str(t)] = stock_price
    #
  if type_put_call == "call":
    difference = max(stock_price - K,0)
  if type_put_call == "put":
    difference = max(K - stock_price,0)
  table.at[index,'difference'] = difference
  #
  table.at[index,'Value'] = difference
  table.at[index,'V'+str(N)] = difference

for t in range (N-1,-1,-1):
  for index,col in table.iterrows():
    #
    index_char = index[t]
    #
    if index_char == "u":
      change_char = "d"
      new_index = index[0:t] + change_char + index[t+1:]
      option_price = p_up * table.loc[index,'V'+str(t+1)] + table.loc[new_index,'V'+str(t+1)] * p_down
      if type_put_call == "call": option_price = max(0, option_price, table.loc[index,'X'+str(t)] - K)
      if type_put_call == "put": option_price = max(0, option_price, K - table.loc[index,'X'+str(t)])
    if index_char == "d":
      change_char = "u"
      new_index = index[0:t] + change_char + index[t+1:]
      option_price = p_up * table.loc[new_index,'V'+str(t+1)] + table.loc[index,'V'+str(t+1)] * p_down
      if type_put_call == "call":
          option_price = max(0, option_price, table.loc[index,'X'+str(t)] - K)
      if type_put_call == "put":
          option_price = max(0, option_price, K - table.loc[index,'X'+str(t)])

    table.at[index,'V'+str(t)] = option_price
    V0 = option_price

for index,col in table.iterrows():
  for t in range (N,0,-1):
    #
    if (table.loc[index,'X'+str(t)] - table.loc[index,'X'+str(t-1)]) != 0:
      hedging_strategy = (table.loc[index,'V'+str(t)] - table.loc[index,'V'+str(t-1)]) / (table.loc[index,'X'+str(t)] -
table.loc[index,'X'+str(t-1)])
```

```
      table.at[index,'HS'+str(t)] = hedging_strategy

  return table

X0=95
K=90
T= 1 # year
r= 0  #
u=1.17 #(Group 7)
d =1/u
N = 5

# ------------------- Q1  ------------------------
function(X0, K, T, r, u, d, N, "put")

# The price of the option is:
# V0 : $11.03
```

The price of the option is:
V0 : $11.03

b) Is there any point time where we, as buyers of the option, benefit from early exercise?
=> Given the computed values above, there is no moment in time when the buyer can gain from buying options early because the option's value rises as it approaches age.

c) Explain whether your previous answer is (or is not) always the case.
=> In real markets, the feature of little gain from early exercise isn't usually true. When the option is fully in the money and close to expiry, its time value is insignificant, a buyer (or trader) would gain from executing the call option early. Another motivation for early exercise is that the buyer will then be able to take advantage of a potential dividend on the underlying stock if the option is exercised early.

Q3->

A. Value of the Option :

Code snippet

```
import numpy as np
import pandas as pd
import itertools
def function(X0, K, T, r, u, d, N, type_put_call = "call", barrier_level ="0"):

  print("Options Calculator:")
  print("X0 : %2d, K : %2d, u : %5.2f, d : %5.4f, N : %2d" % (X0, K,u,d,N))
  p_star = (1-d)/(u-d)
  p_up = p_star
  p_down = 1 - p_star
  print("p* : %5.4f" % (p_star))

  columns=['Value','difference']+['X'+str(i) for i in range(N+1)]+['V'+str(i) for i in range(N+1)]+['HS'+str(i) for i in
range(1,N+1)]
  table = pd.DataFrame(0.0,index= list(itertools.product("ud", repeat=N)),columns=columns)
  table.index=[''.join(i) for i in table.index]
  table['X0']=float(X0)
  print(table)
  for index,col in table.iterrows():
    base_price = X0
    for t in range (1,N+1):
      #
```

```
    i = index[0:t].count("u")
    j = t - i
    stock_price = base_price * u**i * d**j
    table.at[index,'X'+str(t)] = stock_price
  if type == "knockout":
      if type_put_call == "call":
        if stock_price >= 130:
          stock_price = 0
          base_price = 0
      if type_put_call == "put":
        if stock_price <= barrier_level:
          stock_price = 0
          base_price = 0
  table.at[index,'X'+str(t)] = stock_price
    #
  if type_put_call == "call":
    difference = max(stock_price - K,0)
  if type_put_call == "put":
    difference = max(K - stock_price,0)
  table.at[index,'difference'] = difference
  #
  table.at[index,'Value'] = difference
  table.at[index,'V'+str(N)] = difference

for t in range (N-1,-1,-1):
  for index,col in table.iterrows():
    #
    index_char = index[t]
    #
    #
    if index_char == "u":
      change_char = "d"
      new_index = index[0:t] + change_char + index[t+1:]
      option_price = p_up * table.loc[index,'V'+str(t+1)] + table.loc[new_index,'V'+str(t+1)] * p_down
    if index_char == "d":
      change_char = "u"
      new_index = index[0:t] + change_char + index[t+1:]
      option_price = p_up * table.loc[new_index,'V'+str(t+1)] + table.loc[index,'V'+str(t+1)] * p_down

    table.at[index,'V'+str(t)] = option_price
    V0 = option_price
```

```
 for index,col in table.iterrows():
   for t in range (N,0,-1):
     #
     if (table.loc[index,'X'+str(t)] - table.loc[index,'X'+str(t-1)]) != 0:
       hedging_strategy = (table.loc[index,'V'+str(t)] - table.loc[index,'V'+str(t-1)]) / (table.loc[index,'X'+str(t)] -
table.loc[index,'X'+str(t-1)])

       table.at[index,'HS'+str(t)] = hedging_strategy

 return table
```

Value of call option is calculated by
(Sigma H(ω)*probabilities) = 17*5 *0.4608^3*(1-0.4608)^2 = ***2.418***
Refer table attached below :

ω	X0(ω)	X1(ω)	X2(ω)	X3(ω)	X4(ω)	X5(ω)	H(ω)
(uuuuu)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uuuud)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uuudu)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uuudd)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uuduu)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uudud)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uuddu)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uuddd)	100	117.00	129.87	0(KO)	0.00	0.00	0
(uduuu)	100	117.00	100.00	117.00	129.87	0(KO)	0
(uduud)	100	117.00	100.00	117.00	129.87	0(KO)	0
(ududu)	100	117.00	100.00	117.00	100.00	117.00	17
(ududd)	100	117.00	100.00	117.00	100.00	85.47	0
(udduu)	100	117.00	100.00	85.47	100.00	117.00	17
(uddud)	100	117.00	100.00	85.47	100.00	85.47	0
(udddu)	100	117.00	100.00	85.47	77.00	85.47	0
(uddddd)	100	117.00	100.00	85.47	77.00	69.37	0
(duuuu)	100	85.47	100.00	117.00	129.87	0(KO)	0
(duuud)	100	85.47	100.00	117.00	129.87	0(KO)	0
(duudu)	100	85.47	100.00	117.00	100.00	117.00	17
(duudd)	100	85.47	100.00	117.00	100.00	85.47	0
(duduu)	100	85.47	100.00	85.47	100.00	117.00	17
(dudud)	100	85.47	100.00	85.47	100.00	85.47	0
(duddu)	100	85.47	100.00	85.47	77.00	85.47	0
(duddd)	100	85.47	100.00	85.47	77.00	69.37	0
(dduuu)	100	85.47	77.00	85.47	100.00	117.00	17
(dduud)	100	85.47	77.00	85.47	100.00	85.47	0
(ddudu)	100	85.47	77.00	85.47	77.00	85.47	0
(ddudd)	100	85.47	77.00	85.47	77.00	69.37	0
(ddduu)	100	85.47	77.00	69.37	77.00	85.47	0
(dddud)	100	85.47	77.00	69.37	77.00	69.37	0
(ddddu)	100	85.47	77.00	69.37	62.49	69.37	0
(ddddd)	100	85.47	77.00	69.37	62.49	56.30	0

B. **Which option is more expensive: the European call, or the UAO European call?**

Solution : The UAO option is cheaper than the vanilla option because once the price of option / underlying asset raises beyond the barrier value, a knocked out event takes place making that option or underlying asset worthless (ceasing to exist). This is not the case with Vanilla options since the buyer can still exercise the option regardless of its price and vanilla options value do not become 0 under any such circumstances (where value = 0 because of a knockout)

C. **What is the advantage of the up-and-out European call option?**

Solution : UAO options need lower cash structure as compared against plain-vanilla options. Requirement of lower capital means losses will be smaller in case option value becomes zero. It also means larger gains when the option goes in right direction

UAO also provides a low - priced way to hedge positions. It is because it's generally inexpensive that vanilla options

There is more liberty in defining the construct of UAO options as they are generally OTC, hence very customizable as per requirements. This is not the case with Vanilla options since they are traded in an Exchange and are more sophisticated & standard

D. **There is another exotic option called an Up-and-In (UAI) European Call option. Without running a binomial tree, what is the price of UAI with the same strike and maturity as the European call, and the same strike, maturity, and barrier as the UAO. Assume the current stock price is below the barrier. (Hint: Think of a parity using the European call, UAI, and UAO).**

Solution : Up-and-In (UAI) options allows hedging in the case when the strike price is more than the barrier - it is triggered when the price of the underlying asset or option reaches the barrier before maturity. The long positions of UAO and UAI correspond to the long positions of equivalent vanilla options regardless of the spot price behavior related to the barrier level.

Therefore

Vanilla Option = UAO + UAI Options
=> $57.3 (Calculated) = $2.48 (obtained in Q3.a) + UAI

Hence UAI = $57.3 - $2.48 = $54.9

References :

1 Understanding the Binomial Option Pricing Model www.investopedia.com/

2 https://www.linkedin.com/pulse/python-implementation-binomial-stock-option-pricing-sheikh-pancham/

3 What is a Vanilla Option? https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/vanilla-option/

4 Up-and-In Option https://www.investopedia.com/terms/u/up-and-inoption.asp

5 **The Derivatives Academy,** *Maxime de Bellefroid (2017 January)***,** Barrier Options https://bookdown.org/maxime_debellefroid/MyBook/barrier-options.html