Name: Udandarao Sai Sandeep
Roll Number: 180123063
Dept.: Mathematics and Computing

Q1. The standard brute force solution for finding the lookback option involves traversing over all known paths, hence leading to exponential time complexities (2^M). Python was able to calculate the initial lookback option prices only when **M ≤ 20**. Higher values of M exceeded computational capacities of Jupyter Python notebook. The formula used for calculating the option price is as follows:

$$\boldsymbol{H(0) = \frac{1}{e^{rT}} \sum_{over\ all\ paths} p^{ups}(1-p)^{M-ups} f(S_{max})}$$

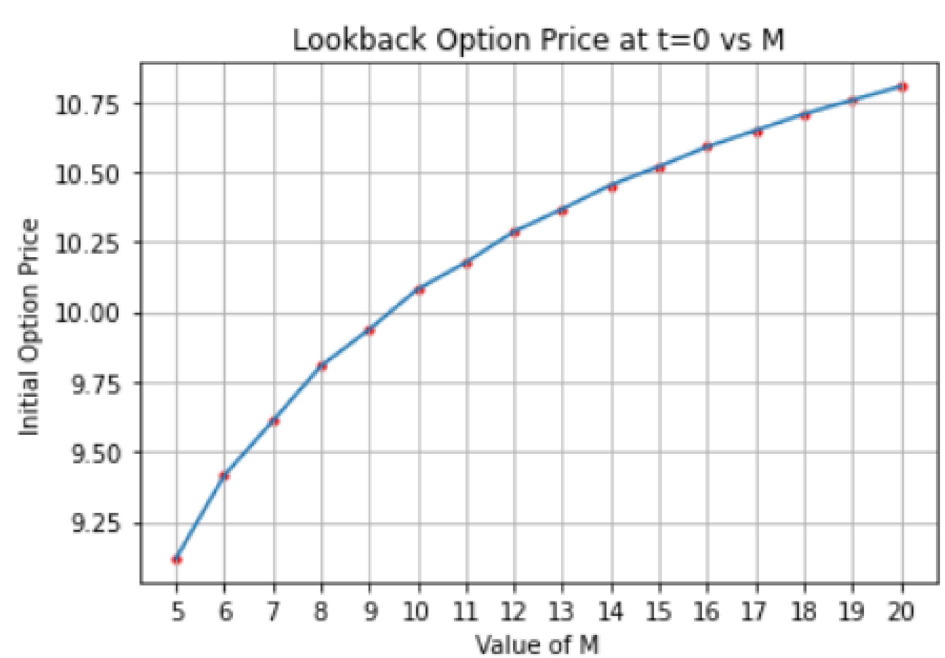
$\boldsymbol{ups}$ $represents\ the\ numebr\ of\ ups\ in\ the\ path$
$\boldsymbol{S_{max}}$ $represents\ the\ maximum\ stock\ price\ over\ the\ path$
$\boldsymbol{f}$ $represents\ the\ payoff:$
$for\ lookback\ option, f(S_{max}) = S_{max} - S(T)$

The values of initial lookback option prices are as follows:

```
q1(a)
m = 5 ,lookback option price = 9.119298985864685
m = 10 ,lookback option price = 10.080582906831074
m = 15 ,lookback option price = 10.519164595672047
m = 20 ,lookback option price = 10.805118587177477
```

For **Q1(b)**, the initial lookback option prices were calculated for the values of **M** between the **range 5 to 20**. A graph of **Option Price vs M** was plotted out. It can be seen that with **increasing value of M**, the **option price also increasing**. However, the graph followed a **concave downward** pattern (a **negative second order derivative**), indicating that as the value of M increases, the curve will eventually flatten out.

q1(b)



For Q1(c), The value of **M has been set to 5.**

For each value of **t**, the **number of remaining time intervals** and **the number of time intervals that have already occurred** were calculated. All possible case paths till time **t** were taken into consideration.

For each possible case, the maximum stock price up till time **t** and **S(t)** was calculated. Considering this as the initial stock price, the option price was calculated (for the remaining time period).

The results obtained are as follows:

```
For t = 1 :
OCCURRED PATH     LOOKBACK OPTION PRICE
    DDDDD             32.105394
    UDDDD             29.482597
    DUDDD             21.234977
    UUDDD             25.394563
    DDUDD             18.805945
    UDUDD             16.266374
    DUUDD             16.266374
    UUUDD             19.452692
    DDDUD             18.805945
    UDDUD             13.578002
    DUDUD             7.818416
    UUDUD             9.349917
    DDUUD             7.818416
    UDUUD             9.349917
    DUUUD             9.349917
    UUUUD             11.181413
    DDDDU             18.805945
    UDDDU             13.578002
    DUDDU             5.330382
    UUDDU             6.374517
    DDUDU             2.901350
    UDUDU             0.000000
    DUUDU             0.000000
    UUUDU             0.000000
    DDDUU             2.901350
    UDDUU             0.000000
    DUDUU             0.000000
    UUDUU             0.000000
    DDUUU             0.000000
    UDUUU             0.000000
    DUUUU             0.000000
    UUUUU             0.000000
```

```
q1(c)

For t = 0 :
OCCURRED PATH     LOOKBACK OPTION PRICE
                      9.119299

For t = 0.2 :
OCCURRED PATH     LOOKBACK OPTION PRICE
    D                 9.504840
    U                 9.027951

For t = 0.4 :
OCCURRED PATH     LOOKBACK OPTION PRICE
    DD                12.168665
    UD                9.799119
    DU                7.147916
    UU                8.548076

For t = 0.6 :
OCCURRED PATH     LOOKBACK OPTION PRICE
    DDD               17.582063
    UDD               13.712863
    DUD               8.324615
    UUD               9.955271
    DDU               7.148418
    UDU               6.201916
    DUU               6.201916
    UUU               7.416771

For t = 0.8 :
OCCURRED PATH     LOOKBACK OPTION PRICE
    DDDD              25.051229
    UDDD              21.188089
    DUDD              13.071381
    UUDD              15.631852
    DDUD              10.680904
    UDUD              8.003614
    DUUD              8.003614
    UUUD              9.571392
    DDDU              10.680904
    UDDU              6.680843
    DUDU              3.846929
    UUDU              4.600480
    DDUU              3.846929
    UDUU              4.600480
    DUUU              4.600480
    UUUU              5.501639
```

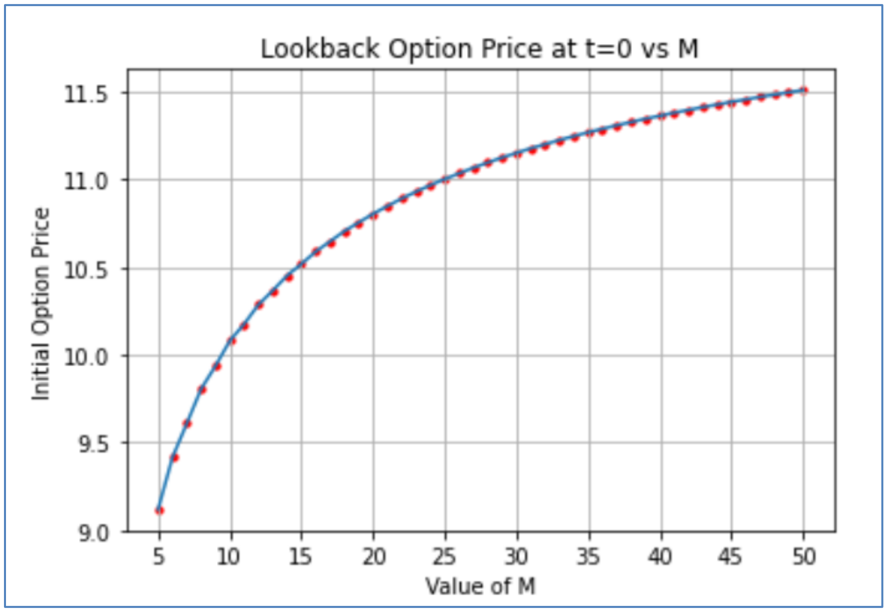
Q2.

For Q2, an efficient **Markov-based binomial algorithm** (as seen in Shreve) was employed to calculate the Lookback Option Prices for various paths. A **recursive algorithm** was used, which keeps track of the **maximum Stock Price** till now, and the **current Stock Price** as different paths are being explored. A **storage data structure (dictionary)** was used to memorize the answers at each stage, in order to avoid repetitive calculations. The time complexity of the program was reduced form **O(2^n)** to **Polynomial Time Complexity.**

The values of Option Prices obtained are as follows:

```
For m = 5 Initial Option Price is 9.11929898586469
For m = 10 Initial Option Price is 10.08058290683101
For m = 15 Initial Option Price is 10.519164595672923
For m = 25 Initial Option Price is 11.003495335646338
For m = 50 Initial Option Price is 11.510862222177268
```

For **Q2**, the initial lookback option prices were calculated for the values of **M** between the **range 5 to 50**. A graph of **Option Price vs M** was plotted out. It can be seen that with **increasing value of M**, the **option price also increasing**. However, the graph followed a **concave downward** pattern (a **negative second order derivative**), indicating that as the value of M increases, the curve will eventually flatten out.



Comparison Between Algorithms used in **Q1** vs **Q2**:

	Straightforward Exponential Algorithm	Markov-based Binomial Algorithm
Values of M it can handle	1 ≤ M ≤ 20	1 ≤ M ≤ 50
Time Complexity	O(2^M)	Polynomial Time Complexity
Time Taken	For m = 5, Time Taken = 0.0016632080078125 For m = 10, Time Taken = 0.014958381652832031 For m = 15, Time Taken = 0.4777700901031494 For m = 20, Time Taken = 17.942022800445557	For m = 5, Time Taken = 0.0016512870788574219 For m = 10, Time Taken = 0.0019898414611816406 For m = 15, Time Taken = 0.00399017333984375 For m = 25, Time Taken = 0.05666470527648926 For m = 50, Time Taken = 3.2304317951202393

Q3.

In order to calculate the Initial European Call Option Price, two different Algorithms were employed. First was the **straightforward exponential** (all possible paths explored) having a time complexity of **$O(2^M)$**. The second algorithm employs a **Markov based efficient binomial algorithm** (involving a **recursive** approach). In this approach, a separate data structure was used to **memorize** already calculated answers to avoid recessive function calls. This algorithm has a time complexity of **$O(M^2)$**.

The values of the Option Prices calculated through both methods are as follows:

```
###############################################################
Case1 : Exponential Calculation
For m = 5 Initial Option Price is 12.163185946764589
For m = 10 Initial Option Price is 12.277327819222997
For m = 15 Initial Option Price is 12.052004991882896
For m = 20 Initial Option Price is 12.174708498955344
For m = 25 Initial Option Price is 12.136745963232963
```

```
###############################################################
Case2 : Markov Based Optimized Calculation
For m = 20 Initial Option Price is 12.17470849895534
For m = 25 Initial Option Price is 12.136745963232972
For m = 50 Initial Option Price is 12.085361510072186
For m = 100 Initial Option Price is 12.104225741732733
For m = 400 Initial Option Price is 12.101330492632442
For m = 900 Initial Option Price is 12.107743235938472
```

Comparison Between Algorithms used in **Q1** vs **Q2**:

	Straightforward Exponential Algorithm	Markov-based Binomial Algorithm
Values of M it can handle	$1 \leq M \leq 25$	$1 \leq M \leq 2000$
Time Complexity	$O(2^M)$	$O(M^2)$
Time Taken	For m = 5, Time Taken = 0.0 For m = 10, Time Taken = 0.0009207725524902344 For m = 15, Time Taken = 0.025980710983276367 For m = 20, Time Taken = 0.8280112743377686 For m = 25, Time Taken = 29.43433952331543	For m = 20, Time Taken = 0.0 For m = 25, Time Taken = 0.0 For m = 50, Time Taken = 0.0010149478912353516 For m = 200, Time Taken = 0.025870561599731445 For m = 400, Time Taken = 0.09474682807922363 For m = 900, Time Taken = 0.561561107635498