```
#Question Number 3
import numpy as np
S_0 = 100
K = 100
T = 1
M_values = [10, 15, 25, 50, 100]
r = 0.08
sigma = 0.2

def efficient_european(S_0, K, M, r, sigma, u, d, p):
    call_list = [0]*(M+1)
    for a in range(M+1):
        call_list[a] = max(S_0*(u**a)*(d**(M-a)) - K, 0)
    for a in range(M):
        for b in range(M-a):
            call_list[b] = ((1-p)*call_list[b] + p*call_list[b+1])*np.exp(-r*T/M)
    call = call_list[0]
    return call

def normal_european(S_0, K, M, r, sigma, u, d, p):
    P = [[[S_0, K]]]
    for a in range(M):
        Q = []
        for el in P[a]:
            Q.append([el[0]*u*p, el[1]*p])
            Q.append([el[0]*d*(1-p), el[1]*(1-p)])
        P.append(Q)
    solution = 0
    for el in P[len(P)-1]:
        solution = solution + max(el[0]-el[1], 0)
    return solution*np.exp(-r*T/M)

for M in M_values:
    dt = T/M
    u = np.exp(sigma*np.sqrt(dt) + (r-0.5*sigma*sigma)*dt)
    d = np.exp(-sigma*np.sqrt(dt) + (r-0.5*sigma*sigma)*dt)
    p = (np.exp(r*dt)-d)/(u-d)
    if M < 25:
        value = normal_european(S_0, K, M, r, sigma, u, d, p)
        print('European Call for M = ', M, 'is', value)
    else:
        print("Normal method cannot give value of M = ", M)
    value = efficient_european(S_0, K, M, r, sigma, u, d, p)
    print('Value of European Call for M = ', M, 'using efficient method is', value
```

```
European Call for M =  10 is 13.193895951751232
Value of European Call for M =  10 using efficient method is 12.2773278192229
European Call for M =  15 is 12.986335667031911
Value of European Call for M =  15 using efficient method is 12.0520049918828
Normal method cannot give value of M =  25
Value of European Call for M =  25 using efficient method is 12.1367459632329
Normal method cannot give value of M =  50
Value of European Call for M =  50 using efficient method is 12.0853615100721
```

```
Normal method cannot give value of M =  100
Value of European Call for M =  100 using efficient method is 12.123047074012
```