In [48]:

```
# Q1 Hadamard Matrix
import numpy as np
def h(n):
    if n==0:
        return np.array([[1]])
    else:
        return np.block([[h(n-1),h(n-1)],
                         [h(n-1),-h(n-1)]])
a = h(4) # for n=4, find H16
b = h(2) # for n=2, find H4
print("H16 is \n",a)
print("H4 is \n",b)
```

```
H16 is 
 [[ 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1]
 [ 1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1  1 -1]
 [ 1  1 -1 -1  1  1 -1 -1  1  1 -1 -1  1  1 -1 -1]
 [ 1 -1 -1  1  1 -1 -1  1  1 -1 -1  1  1 -1 -1  1]
 [ 1  1  1  1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1 -1]
 [ 1 -1  1 -1 -1  1 -1  1  1 -1  1 -1 -1  1 -1  1]
 [ 1  1 -1 -1 -1 -1  1  1  1  1 -1 -1 -1 -1  1  1]
 [ 1 -1 -1  1 -1  1  1 -1  1 -1 -1  1 -1  1  1 -1]
 [ 1  1  1  1  1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1]
 [ 1 -1  1 -1  1 -1  1 -1 -1  1 -1  1 -1  1 -1  1]
 [ 1  1 -1 -1  1  1 -1 -1 -1 -1  1  1 -1 -1  1  1]
 [ 1 -1 -1  1  1 -1 -1  1 -1  1  1 -1 -1  1  1 -1]
 [ 1  1  1  1 -1 -1 -1 -1 -1 -1 -1 -1  1  1  1  1]
 [ 1 -1  1 -1 -1  1 -1  1 -1  1 -1  1  1 -1  1 -1]
 [ 1  1 -1 -1 -1 -1  1  1 -1 -1  1  1  1  1 -1 -1]
 [ 1 -1 -1  1 -1  1  1 -1 -1  1  1 -1  1 -1 -1  1]]
H4 is 
 [[ 1  1  1  1]
 [ 1 -1  1 -1]
 [ 1  1 -1 -1]
 [ 1 -1 -1  1]]
```

In [108…

```
# Q2. given an even integer N, generate a matrix of order NxN
N = 4
c = np.random.randint(-1,2,(N,N//2))
d = np.block([[c,c]])
# calculate the rank of the matrix
np.linalg.matrix_rank(d)
```

```
[[ 1  1]
 [ 1 -1]
 [-1  1]
 [ 0  1]]
```

Out[108… 2

In [168…

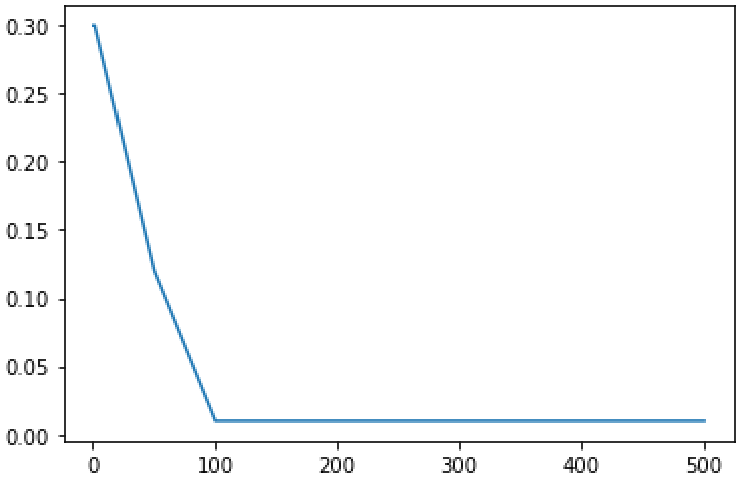
```
# Q5a
import matplotlib.pyplot as plt
# Function to simulate coin tosses
def toss():
    t = np.random.randint(0,100)
    # P(1) = 0.7
    if 70>=t:
        return 1
    else :
        return 0
# Function to calculate average probalilty of tosses
def p_avg(res,N):
    return np.sum(res)/N
# Calculate the error in the probality
```

```
def p_error(p_avg):
    return np.abs(0.7-p_avg)

def run_exp(N):
    res = np.array([]) # Array to store all results
    for i in range(N):
        res = np.append(res,toss())
    p_cap = p_avg(res,N)
    e_M = p_error(p_cap)
    print("For M =",N)
    print("average probability is :",p_cap)
    print("error :", e_M)
    return e_M

M = np.array([1,2,50,100,500])
e = np.array([])
for i in M:
    r = run_exp(i)
    e = np.append(e,r)
plt.plot(M,e)
plt.show()
```

```
For M = 1
average probability is : 1.0
error : 0.30000000000000004
For M = 2
average probability is : 1.0
error : 0.30000000000000004
For M = 50
average probability is : 0.58
error : 0.12
For M = 100
average probability is : 0.71
error : 0.010000000000000009
For M = 500
average probability is : 0.71
error : 0.010000000000000009
```


0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
100
200
300
400
500