AIM :

Plot the probability of various macrostates in "Coin-Tossing Experiment" VS no of heads with 4,8,16 coins etc.

Theory :

1. In general, for 'n' number of coins, there will be 2^n number of Microstates.

2. No of microstates associated with the particular macrostate (say p heads) = nCp = (n!)/((p!)*(n-p)!)

Step-1 : Import necessary libraries

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
from math import comb
```

Step-2 : Take input of "no of coins flipped"

In [2]:
```
n = int(input("Enter the number of coins : "))
```

```
Enter the number of coins : 100
```

Step-3 : Calculate total number of microstates (using above mentioned formula)

In [3]:
```
nom = 2**n
print("Total number of microstates = ",nom)
```

```
Total number of microstates =  1267650600228229401496703205376
```

Step-4 : Find all the macrostates & their respective probabilities

In [4]:
```
#for 'n' coins, there will be 'n+1' macrostates
nh = np.arange(n+1)
print("Number of heads (macrostates) = ",nh)

ps = [] #Defining an empty array for probabilities
for j in nh:
    ns = comb(n,j) #ns = number of possible microstates for a particular macrostate
    psi = ns/nom   #psi = probability for a given macrostate
    ps.append(psi)
print("Respective Probabilities = ",ps)
```

```
Number of heads (macrostates) =  [  0   1   2   3   4   5   6   7   8   9  10  11  12
13  14  15  16  17
  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
```

```
  90  91  92  93  94  95  96  97  98  99 100]
Respective Probabilities =  [7.888609052210118e-31, 7.888609052210118e-29, 3.9048614808
440084e-27, 1.275588083742376e-25, 3.093301103075262e-24, 5.939138117904503e-23, 9.4036
35353348797e-22, 1.2627738903068384e-20, 1.4679746474816996e-19, 1.5005963063146263e-1
8, 1.3655426387463099e-17, 1.1172621589742536e-16, 8.286361012392381e-16, 5.60922899300
4073e-15, 3.4857351599382454e-14, 1.998488158364594e-13, 1.0616968341311906e-12, 5.2460
31415707059e-12, 2.4190033750204773e-11, 1.0439909302719954e-10, 4.2281632676015815e-1
0, 1.6107288638482216e-09, 5.78398092018225e-09, 1.9615239642357197e-08, 6.293222718589
6e-08, 1.9131397064512386e-07, 5.518672230147804e-07, 1.5125249815960647e-06, 3.9433687
020183116e-06, 9.790432639493739e-06, 2.3170690580135184e-05, 5.232091421320847e-05, 0.
00011281697127223077, 0.00023247133474277857, 0.0004581052772872401 4, 0.000863855665741
6528, 0.001559739396477984 2, 0.0026979276047186754, 0.00447287997624412, 0.007110732269
92655, 0.010843866711637987, 0.015869073236543397, 0.022292269546572867, 0.030068642644
214563, 0.03895255978909614, 0.048474296626430755, 0.05795839814029764, 0.0665904999909
8027, 0.07352701040670738, 0.07802866410507722, 0.07958923738717877, 0.0780286641050772
2, 0.07352701040670738, 0.06659049999098027, 0.05795839814029764, 0.048474296626430755,
0.03895255978909614, 0.030068642644214563, 0.022292269546572867, 0.015869073236543397,
0.010843866711637987, 0.00711073226992655, 0.00447287997624412, 0.0026979276047186754,
0.001559739396477984 2, 0.0008638556657416528, 0.0004581052772872401 4, 0.000232471334742
77857, 0.00011281697127223077, 5.232091421320847e-05, 2.3170690580135184e-05, 9.7904326
39493739e-06, 3.9433687020183116e-06, 1.5125249815960647e-06, 5.518672230147804e-07, 1.
9131397064512386e-07, 6.2932227185896e-08, 1.9615239642357197e-08, 5.78398092018225e-0
9, 1.6107288638482216e-09, 4.2281632676015815e-10, 1.0439909302719954e-10, 2.4190033750
204773e-11, 5.246031415707059e-12, 1.0616968341311906e-12, 1.998488158364594e-13, 3.485
7351599382454e-14, 5.609228993004073e-15, 8.286361012392381e-16, 1.1172621589742536e-1
6, 1.3655426387463099e-17, 1.5005963063146263e-18, 1.4679746474816996e-19, 1.2627738903
068384e-20, 9.403635353348797e-22, 5.939138117904503e-23, 3.093301103075262e-24, 1.2755
```

Step-5 : Print Table (for visual aid)

In [5]:

```
print("n(h)   P(h)")
print("-----------")
for i in range(n+1):
    print(nh[i],"   ",ps[i])
```

```
n(h)   P(h)
-----------
0     7.888609052210118e-31
1     7.888609052210118e-29
2     3.9048614808440084e-27
3     1.275588083742376e-25
4     3.093301103075262e-24
5     5.939138117904503e-23
6     9.403635353348797e-22
7     1.2627738903068384e-20
8     1.4679746474816996e-19
9     1.5005963063146263e-18
10     1.3655426387463099e-17
11     1.1172621589742536e-16
12     8.286361012392381e-16
13     5.609228993004073e-15
14     3.4857351599382454e-14
15     1.998488158364594e-13
16     1.0616968341311906e-12
17     5.246031415707059e-12
18     2.4190033750204773e-11
19     1.0439909302719954e-10
20     4.2281632676015815e-10
21     1.6107288638482216e-09
22     5.78398092018225e-09
23     1.9615239642357197e-08
24     6.2932227185896e-08
25     1.9131397064512386e-07
26     5.518672230147804e-07
27     1.5125249815960647e-06
28     3.9433687020183116e-06
29     9.790432639493739e-06
```

30 2.3170690580135184e-05
31 5.232091421320847e-05
32 0.00011281697127223077
33 0.00023247133474277857
34 0.0004581052772872401 4
35 0.0008638556657416528
36 0.0015597393964779842
37 0.0026979276047186754
38 0.00447287997624412
39 0.0071107322699265 5
40 0.010843866711637987
41 0.015869073236543397
42 0.022292269546572867
43 0.030068642644214563
44 0.03895255978909614
45 0.048474296626430755
46 0.05795839814029764
47 0.06659049999098027
48 0.07352701040670738
49 0.07802866410507722
50 0.07958923738717877
51 0.07802866410507722
52 0.07352701040670738
53 0.06659049999098027
54 0.05795839814029764
55 0.048474296626430755
56 0.03895255978909614
57 0.030068642644214563
58 0.022292269546572867
59 0.015869073236543397
60 0.010843866711637987
61 0.0071107322699265 5
62 0.00447287997624412
63 0.0026979276047186754
64 0.0015597393964779842
65 0.0008638556657416528
66 0.0004581052772872401 4
67 0.00023247133474277857
68 0.00011281697127223077
69 5.232091421320847e-05
70 2.3170690580135184e-05
71 9.790432639493739e-06
72 3.9433687020183116e-06
73 1.5125249815960647e-06
74 5.518672230147804e-07
75 1.9131397064512386e-07
76 6.2932227185896e-08
77 1.9615239642357197e-08
78 5.78398092018225e-09
79 1.6107288638482216e-09
80 4.2281632676015815e-10
81 1.0439909302719954e-10
82 2.4190033750204773e-11
83 5.246031415707059e-12
84 1.0616968341311906e-12
85 1.998488158364594e-13
86 3.4857351599382454e-14
87 5.609228993004073e-15
88 8.286361012392381e-16
89 1.1172621589742536e-16
90 1.3655426387463099e-17
91 1.5005963063146263e-18
92 1.4679746474816996e-19
93 1.2627738903068384e-20
94 9.403635353348797e-22
95 5.939138117904503e-23
96 3.093301103075262e-24
97 1.275588083742376e-25
98 3.9048614808440084e-27

```
99     7.888609052210118e-29
100     7.888609052210118e-31
```

Step-6 : Plot the required graph

In [6]:

```
plt.figure(figsize=(15,8))
fontji = {'family':'serif','size':20}
fontji2 = {'family':'serif','size':30}

plt.plot(nh,ps,"o-r",lw="5",ms="10")
plt.xlabel("Number of Heads",color="green",fontdict=fontji)
plt.ylabel("Respective Probabilities",color="green",fontdict=fontji)
plt.title("Probability VS Number of Heads",color="blue",fontdict=fontji2)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
```

Out[6]:

```
(array([-0.01,  0.  ,  0.01,  0.02,  0.03,  0.04,  0.05,  0.06,  0.07,
         0.08,  0.09]),
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```