Bhartiya Vidya Bhavan's
Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058-India
(Autonomous College Affiliated to University of Mumbai)

NAME:	Vaishnavi Bhagawan Borkar
UID:	2021300016
SUBJECT	Data Analysis and Algorithm
EXPERIMENT NO:	Experiment 1-A
DATE OF PERFORMANCE	23-01-23
AIM:	To implement the various linear and non-linear functions from the given list of functions, implement at least 10, print the input-output table and plot the graphs. Also, write your observations.
THEORY:	A function in the context of this experiment is a mathematical expression that gives certain outputs based on the inputs provided. A proper understanding of functions in general and their nature would help us while dealing with the time complexities of algorithms. It would help us to compare two or more algorithms in the process of determining the most efficient one.
ALGORITHM:	**[A] For log functions**: Call the log function from the 'math.h' header file. For natural log, the function is 'log' itself. For log of specific base we just have to write the base number along with log. **[B] For other functions:** Simply call the required function from the math header file and return the obtained value.

PROGRAM:	`#include <stdio.h>`
	`#include <math.h>`
	`int n;`
	`void f1()`

```
{
    int ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = n;
        printf("%d %d", n, ans[n]);
        printf("\n");
    }
}
void f2()
{
    int ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = n * n * n;
        printf("%d %d", n, ans[n]);
        printf("\n");
    }
}
void f3()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = pow(2, n);
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
void f4()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = pow(1.5, n);
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double f5()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
```

```
    {
        if (n == 0)
        {
            continue;
        }
        ans[n] = log(n);
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double f6()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = log10(n);
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double f7()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = sqrt(log2(n));
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double f8()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = n * pow(2, n);
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double f9()
{
    double ans[100];
```

```
    for (n = 0; n <= 100; n++)
    {
        ans[n] = exp(n);
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double f10()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = pow(2, log2(n));
        printf("%d %.2f", n, ans[n]);
        printf("\n");
    }
}
double fac(int i)
{
    if (i >= 1)
        return i * fac(i - 1);
    else
        return 1;
}
double f11()
{
    double ans[100];
    for (n = 0; n <= 100; n++)
    {
        ans[n] = fac(n);
        printf("%.2f", ans[n]);
        printf("\n");
    }
}
int main()
{

printf("Function:\n1.n\n2.n^3\n3.2^n\n4.(3/2)^n\n5.ln(n)\n6.lg(n)\
n7.square root lgn\n8.n*2^n\n9.e^n\n10.2^(log(n))\n11.n!");
    printf("\nEnter your choice:");
    int ch;
    scanf("%d", &ch);
    if (ch == 1)
```

```
    {
        f1();
    }
    else if (ch == 2)
    {
        f2();
    }
    else if (ch == 3)
    {
        f3();
    }
    else if (ch == 4)
    {
        f4();
    }
    else if (ch == 5)
    {
        f5();
    }
    else if (ch == 6)
    {
        f6();
    }
    else if (ch == 7)
    {
        f7();
    }
    else if (ch == 8)
    {
        f8();
    }
    else if (ch == 9)
    {
        f9();
    }
    else if (ch == 10)
    {
        f10();
    }
    else if (ch == 11)
    {
        f11();
    }
```

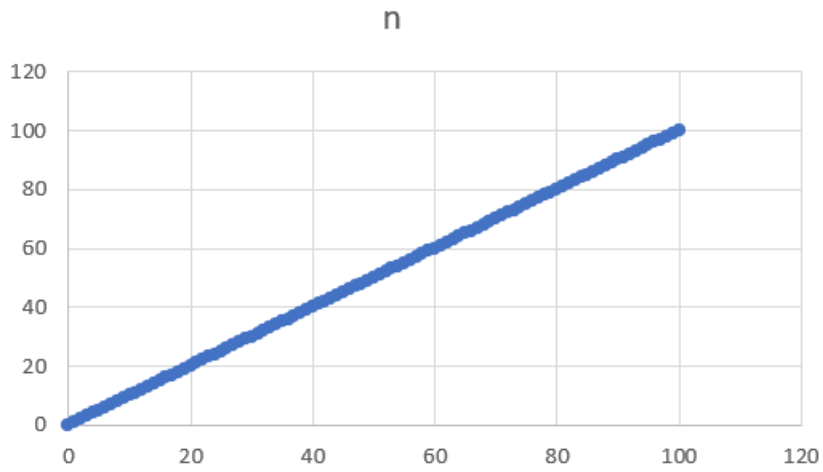
```
}
```

RESULT:

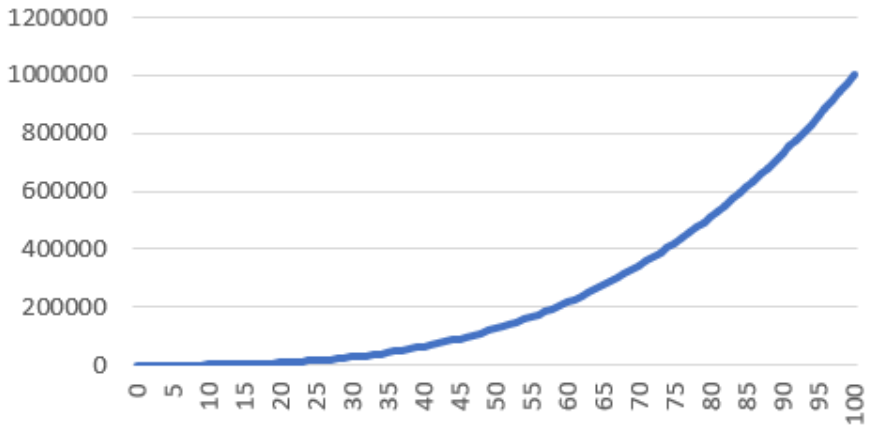
```
Enter your choice:5
1 0.00
2 0.69
3 1.10
4 1.39
5 1.61
6 1.79
7 1.95
8 2.08
9 2.20
10 2.30
11 2.40
12 2.48
13 2.56
14 2.64
15 2.71
16 2.77
17 2.83
18 2.89
19 2.94
20 3.00
21 3.04
22 3.09
23 3.14
24 3.18
25 3.22
26 3.26
27 3.30
28 3.33
29 3.37
30 3.40
31 3.43
32 3.47
33 3.50
34 3.53
```

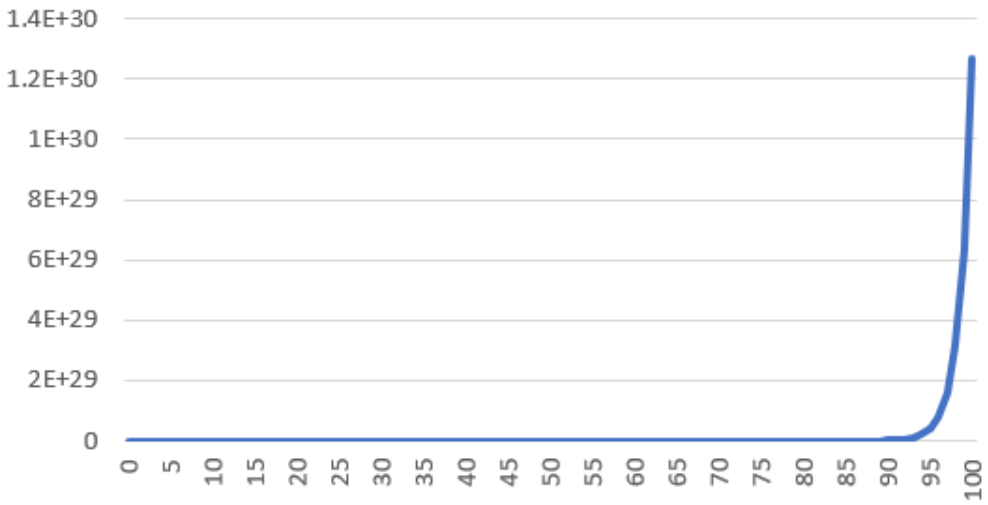
```
35 3.56
36 3.58
37 3.61
38 3.64
39 3.66
40 3.69
41 3.71
42 3.74
43 3.76
44 3.78
45 3.81
46 3.83
47 3.85
48 3.87
49 3.89
50 3.91
51 3.93
52 3.95
53 3.97
54 3.99
55 4.01
56 4.03
57 4.04
58 4.06
59 4.08
60 4.09
61 4.11
62 4.13
63 4.14
64 4.16
65 4.17
92 4.52
93 4.53
94 4.54
95 4.55
96 4.56
```

GRAPH:

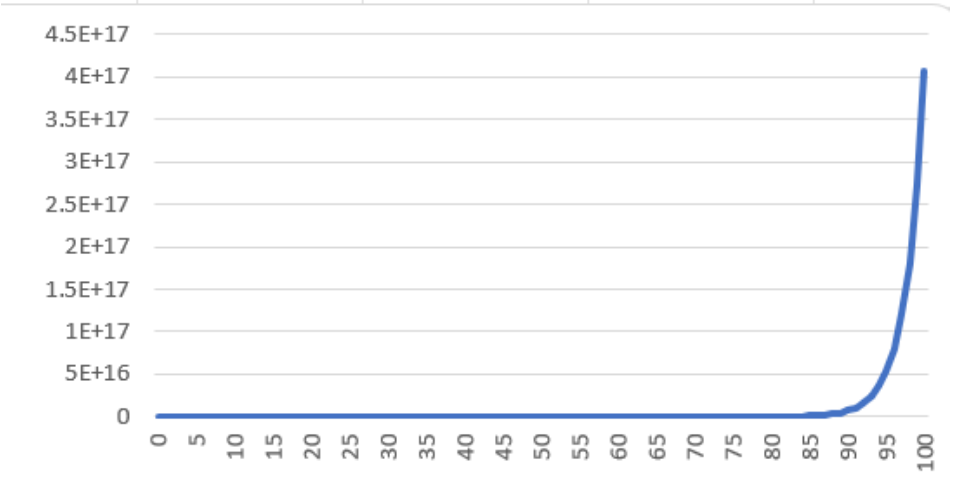


This a n function which gives a linear graph i.e as n increases the value of function increases linearly.

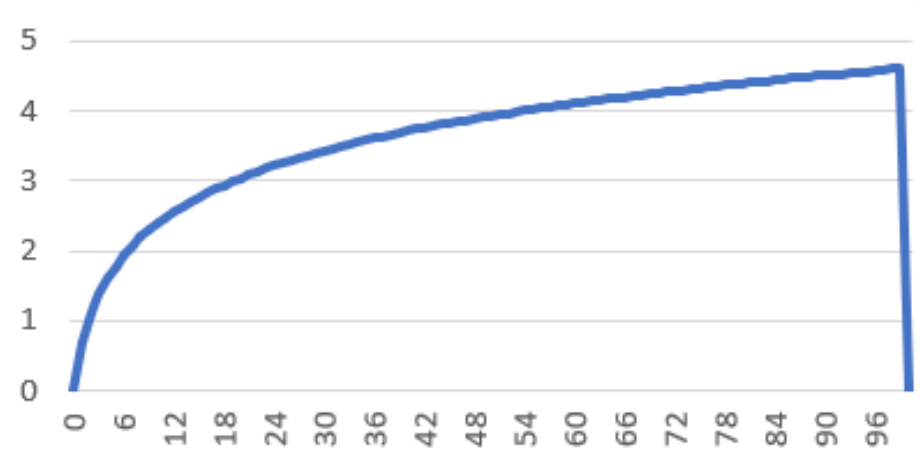




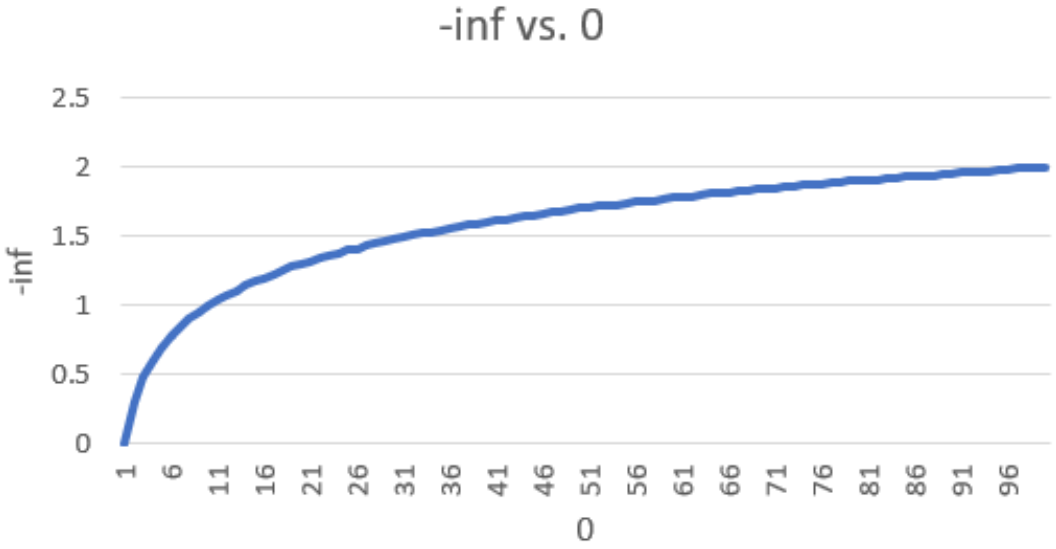
This a 2^n function which gives a exponential graph i.e as n increasing the value of function increases exponentially.



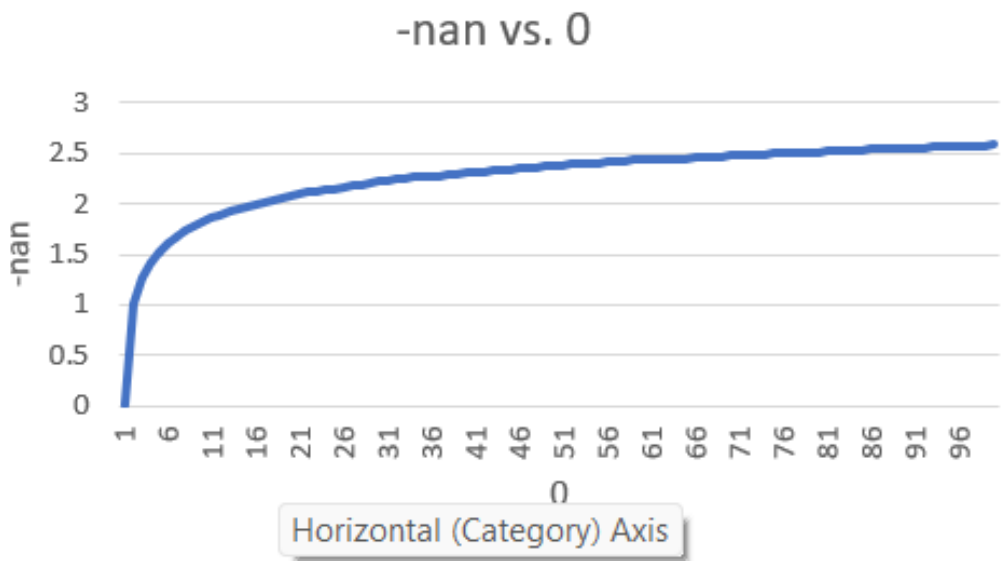
This a (3/2)^n function which gives an exponential graph i.e as n increasing the value of function increases exponentially.



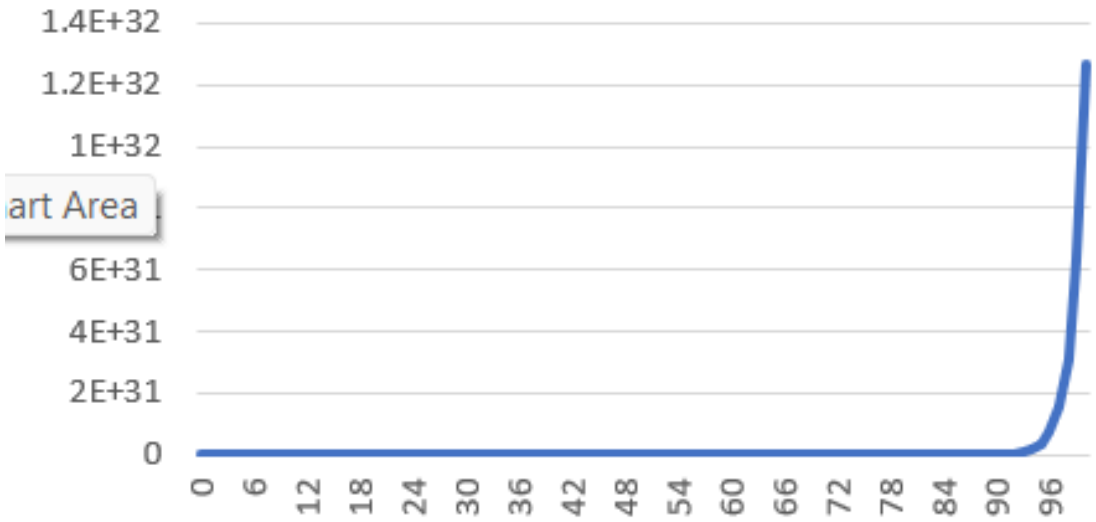
This a ln(n) function which gives an exponential graph i.e as n increasing the value of function increases exponentially



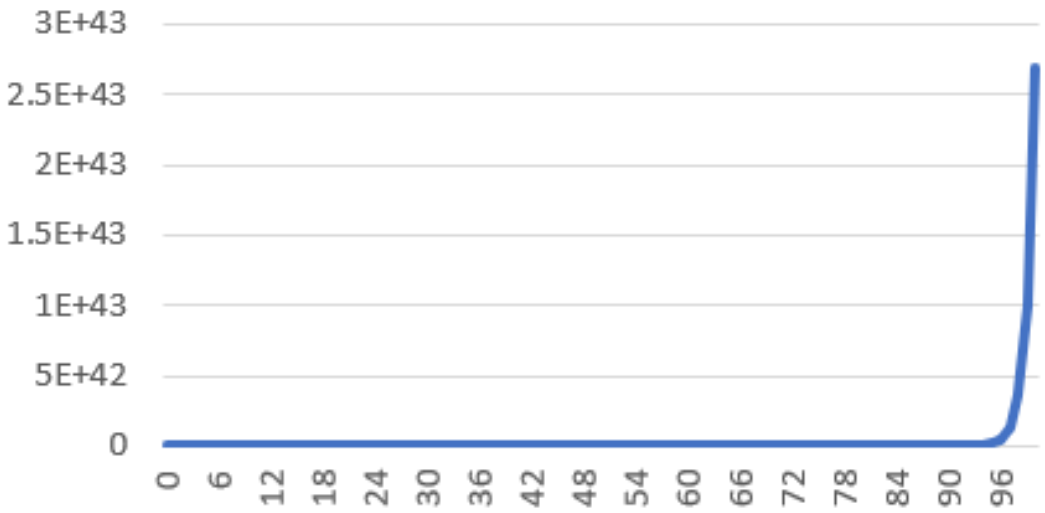
This a lg(n) function which gives a logarithmic graph i.e as n increasing the value of function increases logarithmically.



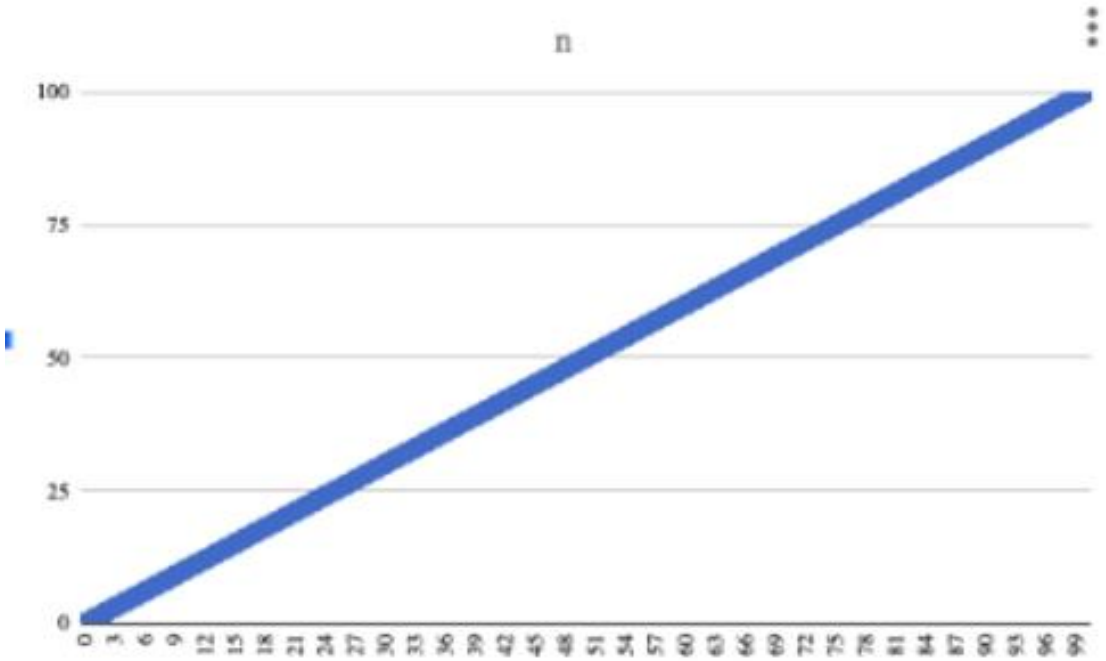
This a square root lgn function which gives a logarithmic graph i.e as n increasing the value of function increases logarithmically.



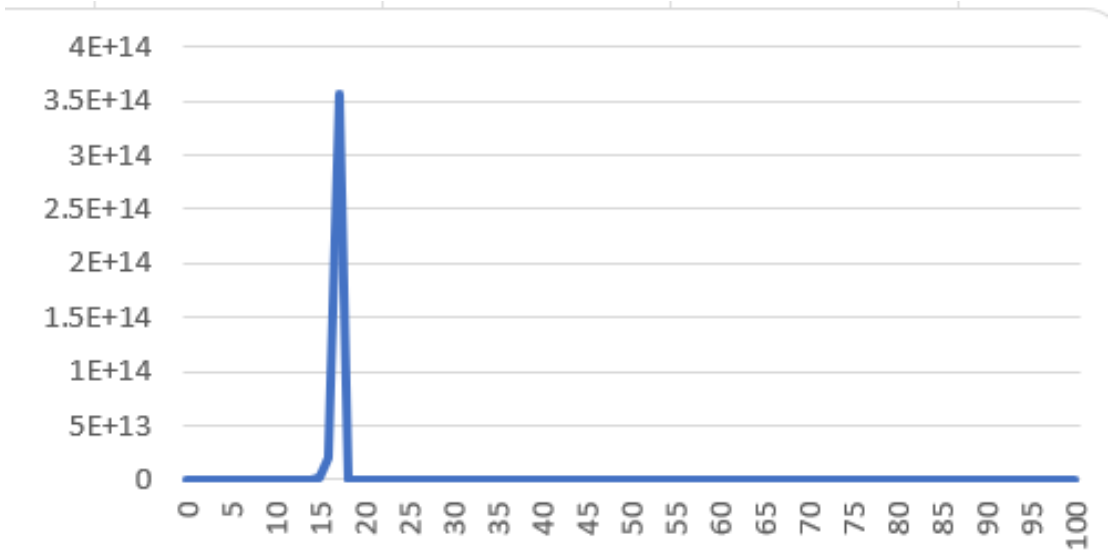
This a n*2^n function which gives a exponential graph i.e as n increasing the value of function increases exponentially.



This an e^n function which gives an exponential graph i.e as n increasing the value of function increases exponentially.



This a 2^(log(n)) function which gives a linear graph i.e as n increases the value of function increases linearly.



This a n! function which gives an exponential graph i.e as n increasing the value of function increases exponentially.

CONCLUSION:	By performing the experiment, I understood the algorithm behind coding and the use of graph plotting.