Name	Aditi Nilesh Bhutada
UID no.	2021700009
Experiment No.	1 A

AIM:	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.
PROGRAM	
THEORY:	A function is a relation between a set of inputs and a set of permissible outputs with the property that each input is related to exactly one output. Let A & B be any two non-empty sets; mapping from A to B will be a function only when every element in set A has one end, only one image in set B.
INPUT:	We take input as n= numbers from 0 to 100
ALGORITHM:	Step 1: Select 10 functions from the given list Step 2: Include math.h from the library to calculate the values of the given functions with the input values Step 3: In the main function declare an integer variable n which represents the input values which will vary from 0 to 100 Step 4: Create 10 functions and store the calculated values in a double variable, pass the variable n as argument and print or return the calculated value Step 5: Now in the main function declare a variable i and initialize a for loop with i=0 to i=100 and increment the value of i with every iteration Step 6: In the for loop call all the 10 functions so that we get the values till last iteration in a tabular form Step 7: Copy paste the output in an excel sheet and plot 2D graphs for every function for n=0 to n=100

PROGRAM:

```
#include <stdio.h>
#include <math.h>

void func1(int n){    //n^3
   int res=n*n*n;
   printf("%d\t",res);
}

void func2(int n){   // ln(n)
   double res=log(n);
    printf("%.3f\t",res);
}

void func3(int n){        // nlg(n)
    double res=n*log2(n);
    printf("%.3f\t",res);
}

void func4(int n){  // 2^ln(n)

    double res = pow(2,log(n));
    printf("%.3f\t",res);
}

void func5(int n){
    double res = pow(log2(n),0.5); // sqrt(log2n)
    printf("%.3f\t",res);
}

void func6(int n){   // log2 n
   double res=log2(n);
    printf("%.3f\t",res);
}

void func7(int n){   // (root 2)^ log2n
    double res = pow(sqrt(2),log2(n));
     printf("%.3f\t",res);
}

void func8(int n){  // ln ln n
    double res = log(log(n));
    printf("%.3f\t",res);
}

void func9(int n){   // n
  printf("%d\t",n);
```

```
}

void func10(int n){   // n^ (lglgn)
   double res = pow(2,2*log2(n));
    printf("%.2f\t",res);
}

int main()
{
   printf("n\t");
printf("Func1\tFunc2\tFunc3\tFunc4\tFunc5\tFunc6\tFunc7\tFunc8\tFunc9\tFunc10\t");
   printf("\n");
   for(int i=0;i<=100;i++){
      printf("%d\t",i);
      func1(i); func2(i);
      func3(i); func4(i);
      func5(i); func6(i);
      func7(i); func8(i);
      func9(i); func10(i);
      printf("\n");
   }
   return 0;
}
```

OUTPUT:

n	Func1	Func2	Func3	Func4	Func5	Func6	Func7	Func8	Func9	Func10
0	0	-inf	-nan	0.000	inf	-inf	0.000	-nan	0	0.00
1	1	0.000	0.000	1.000	0.000	0.000	1.000	-inf	1	1.00
2	8	0.693	2.000	1.617	1.000	1.000	1.414	-0.367	2	4.00
3	27	1.099	4.755	2.141	1.259	1.585	1.732	0.094	3	9.00
4	64	1.386	8.000	2.614	1.414	2.000	2.000	0.327	4	16.00
5	125	1.609	11.610	3.051	1.524	2.322	2.236	0.476	5	25.00
6	216	1.792	15.510	3.462	1.608	2.585	2.449	0.583	6	36.00
7	343	1.946	19.651	3.853	1.676	2.807	2.646	0.666	7	49.00
8	512	2.079	24.000	4.226	1.732	3.000	2.828	0.732	8	64.00
9	729	2.197	28.529	4.586	1.780	3.170	3.000	0.787	9	81.00
10	1000	2.303	33.219	4.933	1.823	3.322	3.162	0.834	10	100.00
11	1331	2.398	38.054	5.270	1.860	3.459	3.317	0.875	11	121.00
12	1728	2.485	43.020	5.598	1.893	3.585	3.464	0.910	12	144.00
13	2197	2.565	48.106	5.917	1.924	3.700	3.606	0.942	13	169.00
14	2744	2.639	53.303	6.229	1.951	3.807	3.742	0.970	14	196.00
15	3375	2.708	58.603	6.534	1.977	3.907	3.873	0.996	15	225.00
16	4096	2.773	64.000	6.833	2.000	4.000	4.000	1.020	16	256.00
17	4913	2.833	69.487	7.127	2.022	4.087	4.123	1.041	17	289.00
18	5832	2.890	75.059	7.415	2.042	4.170	4.243	1.061	18	324.00
19	6859	2.944	80.711	7.698	2.061	4.248	4.359	1.080	19	361.00
20	8000	2.996	86.439	7.976	2.079	4.322	4.472	1.097	20	400.00
21	9261	3.045	92.239	8.251	2.096	4.392	4.583	1.113	21	441.00
22	10648	3.091	98.107	8.521	2.112	4.459	4.690	1.129	22	484.00
23	12167	3.135	104.042	8.788	2.127	4.524	4.796	1.143	23	529.00
24	13824	3.178	110.039	9.051	2.141	4.585	4.899	1.156	24	576.00
25	15625	3.219	116.096	9.311	2.155	4.644	5.000	1.169	25	625.00
26	17576	3.258	122.211	9.567	2.168	4.700	5.099	1.181	26	676.00
27	19683	3.296	128.382	9.821	2.181	4.755	5.196	1.193	27	729.00
28	21952	3.332	134.606	10.071	2.193	4.807	5.292	1.204	28	784.00
29	24389	3.367	140.881	10.319	2.204	4.858	5.385	1.214	29	841.00
			147.207	10.565	2.215	4.907	5.477	1.224	30	900.00

32	32768	3.466	160.000	11.048	2.236	5.000	5.657	1.243	32	1024.00
33	35937	3.497	166.465	11.286	2.246	5.044	5.745	1.252	33	1089.00
34	39304	3.526	172.974	11.522	2.256	5.087	5.831	1.260	34	1156.00
35	42875	3.555	179.525	11.756	2.265	5.129	5.916	1.268	35	1225.00
36	46656	3.584	186.117	11.988	2.274	5.170	6.000	1.276	36	1296.00
37	50653	3.611	192.750	12.218	2.282	5.209	6.083	1.284	37	1369.00
38	54872	3.638	199.421	12.446	2.291	5.248	6.164	1.291	38	1444.00
39	59319	3.664	206.131	12.672	2.299	5.285	6.245	1.298	39	1521.00
40	64000	3.689	212.877	12.896	2.307	5.322	6.325	1.305	40	1600.00
41	68921	3.714	219.660	13.119	2.315	5.358	6.403	1.312	41	1681.00
42	74088	3.738	226.477	13.340	2.322	5.392	6.481	1.318	42	1764.00
43	79507	3.761	233.329	13.559	2.329	5.426	6.557	1.325	43	1849.00
44	85184	3.784	240.215	13.777	2.337	5.459	6.633	1.331	44	1936.00
45	91125	3.807	247.133	13.993	2.343	5.492	6.708	1.337	45	2025.00
46	97336	3.829	254.084	14.208	2.350	5.524	6.782	1.343	46	2116.00
47	103823	3.850	261.066	14.421	2.357	5.555	6.856	1.348	47	2209.00
48	110592	3.871	268.078	14.633	2.363	5.585	6.928	1.354	48	2304.00
49	117649	3.892	275.121	14.844	2.370	5.615	7.000	1.359	49	2401.00
50	125000	3.912	282.193	15.053	2.376	5.644	7.071	1.364	50	2500.00
51	132651	3.932	289.294	15.262	2.382	5.672	7.141	1.369	51	2601.00
52	140608	3.951	296.423	15.468	2.388	5.700	7.211	1.374	52	2704.00
53	148877	3.970	303.580	15.674	2.393	5.728	7.280	1.379	53	2809.00
54	157464	3.989	310.764	15.878	2.399	5.755	7.348	1.384	54	2916.00
55	166375	4.007	317.975	16.082	2.404	5.781	7.416	1.388	55	3025.00
56	175616	4.025	325.212	16.284	2.410	5.807	7.483	1.393	56	3136.00
57	185193	4.043	332.475	16.485	2.415	5.833	7.550	1.397	57	3249.00
58	195112	4.060	339.763	16.685	2.420	5.858	7.616	1.401	58	3364.00
59	205379	4.078	347.076	16.883	2.425	5.883	7.681	1.405	59	3481.00
60	216000	4.094	354.413	17.081	2.430	5.907	7.746	1.410	60	3600.00
61	226981	4.111	361.775	17.278	2.435	5.931	7.810	1.414	61	3721.00
62	238328	4.127	369.160	17.474	2.440	5.954	7.874	1.418	62	3844.00
63	250047	4.143	376.569	17.669	2.445	5.977	7.937	1.421	63	3969.00

for bidder.criteo.com...

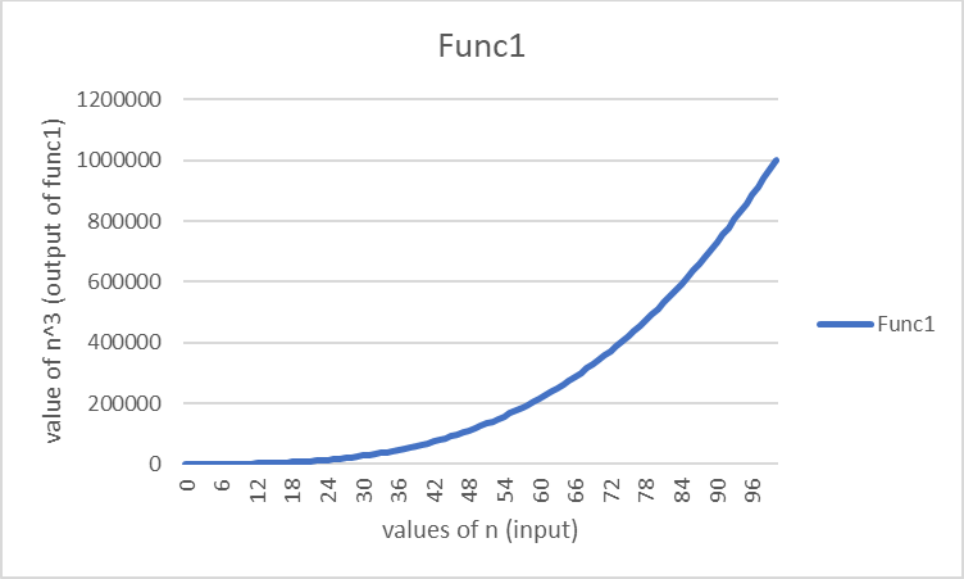
73	389017	4.290	451.857	19.568	2.488	6.190	8.544	1.456	73	5329.00
74	405224	4.304	459.500	19.754	2.492	6.209	8.602	1.460	74	5476.00
75	421875	4.317	467.161	19.939	2.496	6.229	8.660	1.463	75	5625.00
76	438976	4.331	474.842	20.122	2.500	6.248	8.718	1.466	76	5776.00
77	456533	4.344	482.543	20.306	2.503	6.267	8.775	1.469	77	5929.00
78	474552	4.357	490.261	20.488	2.507	6.285	8.832	1.472	78	6084.00
79	493039	4.369	497.999	20.670	2.511	6.304	8.888	1.475	79	6241.00
80	512000	4.382	505.754	20.851	2.514	6.322	8.944	1.478	80	6400.00
81	531441	4.394	513.528	21.031	2.518	6.340	9.000	1.480	81	6561.00
82	551368	4.407	521.319	21.211	2.521	6.358	9.055	1.483	82	6724.00
83	571787	4.419	529.128	21.390	2.525	6.375	9.110	1.486	83	6889.00
84	592704	4.431	536.955	21.568	2.528	6.392	9.165	1.489	84	7056.00
85	614125	4.443	544.798	21.746	2.532	6.409	9.220	1.491	85	7225.00
86	636056	4.454	552.659	21.923	2.535	6.426	9.274	1.494	86	7396.00
87	658503	4.466	560.536	22.099	2.538	6.443	9.327	1.496	87	7569.00
88	681472	4.477	568.430	22.275	2.542	6.459	9.381	1.499	88	7744.00
89	704969	4.489	576.340	22.450	2.545	6.476	9.434	1.502	89	7921.00
90	729000	4.500	584.267	22.624	2.548	6.492	9.487	1.504	90	8100.00
91	753571	4.511	592.209	22.798	2.551	6.508	9.539	1.506	91	8281.00
92	778688	4.522	600.168	22.972	2.554	6.524	9.592	1.509	92	8464.00
93	804357	4.533	608.142	23.145	2.557	6.539	9.644	1.511	93	8649.00
94	830584	4.543	616.131	23.317	2.560	6.555	9.695	1.514	94	8836.00
95	857375	4.554	624.136	23.488	2.563	6.570	9.747	1.516	95	9025.00
96	884736	4.564	632.156	23.660	2.566	6.585	9.798	1.518	96	9216.00
97	912673	4.575	640.192	23.830	2.569	6.600	9.849	1.521	97	9409.00
98	941192	4.585	648.242	24.000	2.572	6.615	9.899	1.523	98	9604.00
99	970299	4.595	656.306	24.170	2.575	6.629	9.950	1.525	99	9801.00
100	1000000	4.605	664.386	24.339	2.578	6.644	10.000	1.527	100	10000.00

...Program finished with exit code 0

GRAPHS:

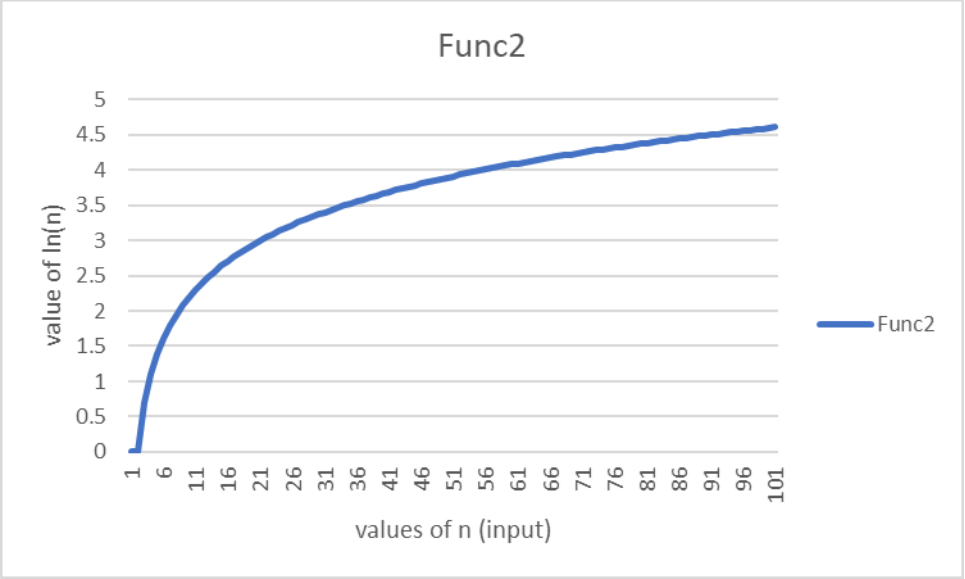
1. Func1= n^3

- Here we see that the graph increases gradually from 0 for n=0 to 1000000 for n=100, having a positive slope. The slope initially is very less and increases further till n=100



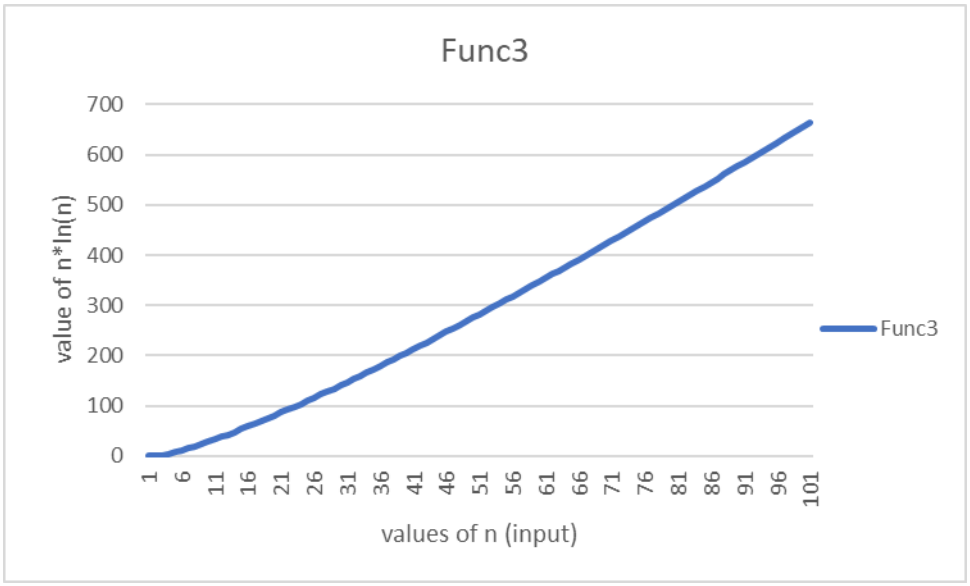
2. Func2= ln(n) with base e

- Here the value of ln0 is undefined. It has a slope=1/x. The values in the graph increases more and has a greater slope from n=0 to 10 and later the slope decreases as the values increases by very small amount



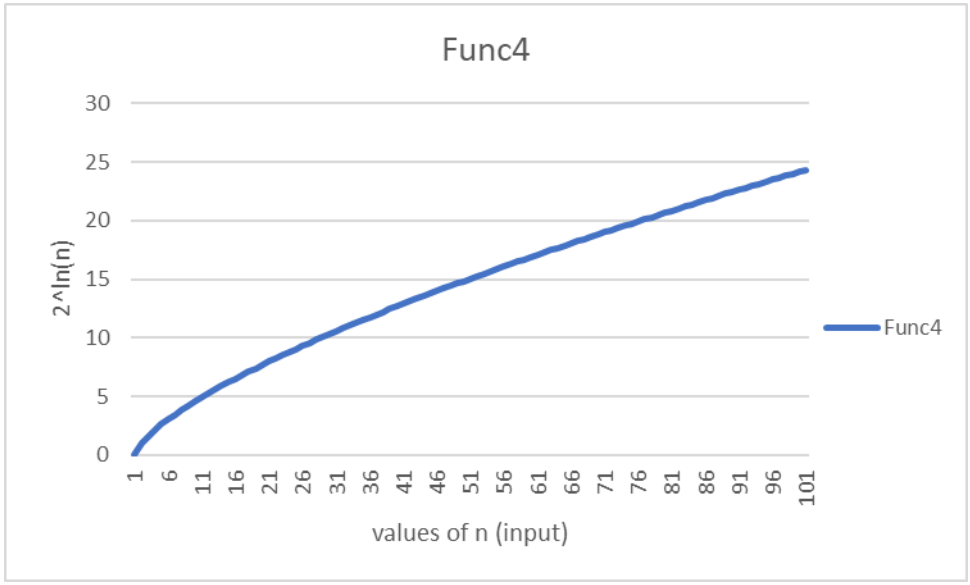
3. Func3= n*ln(n)

- Here the graph starts from 0, has a positive slope and it tends to be constant from n=5 to n=100 but has a very minimal increase



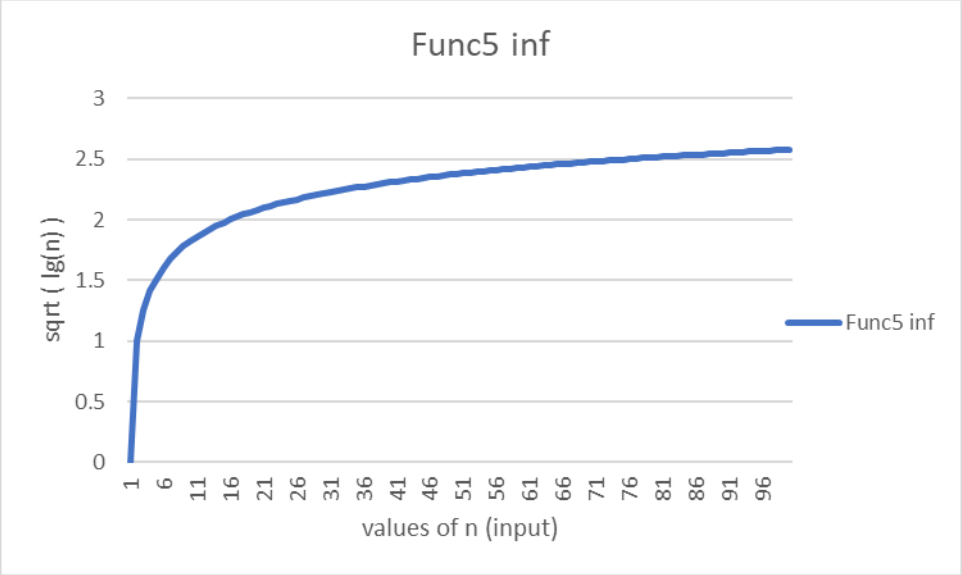
4. Func4= 2^ln(n)

- It is an increasing graph, the slope of the graph was greater from around n=0 to n=10 and decreases with a minimal change



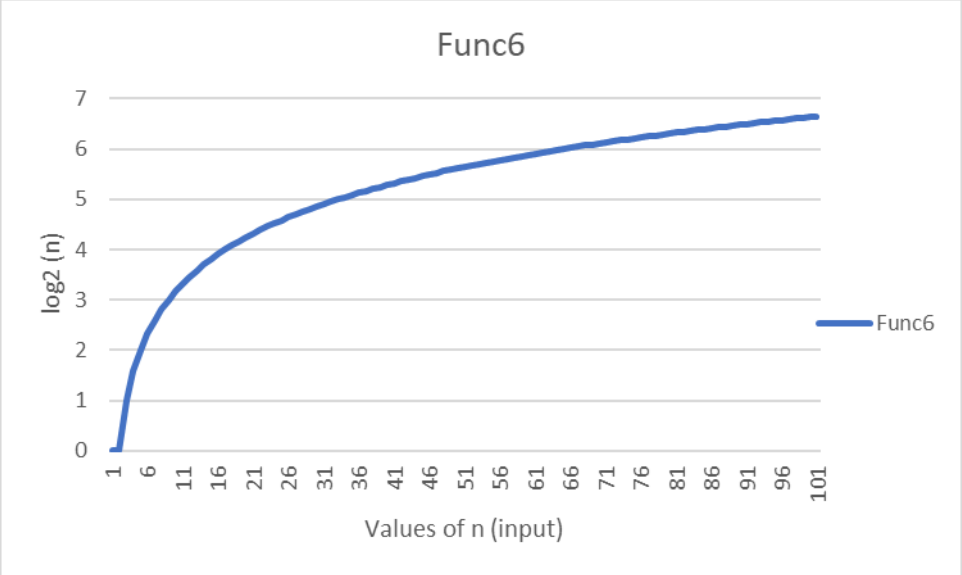
5. Func5= sqrt (log2(n))

- The value of the function increases drastically for the initial inputs till around n=8 and then the values of function and the slope bends and decreases as the value of n increases



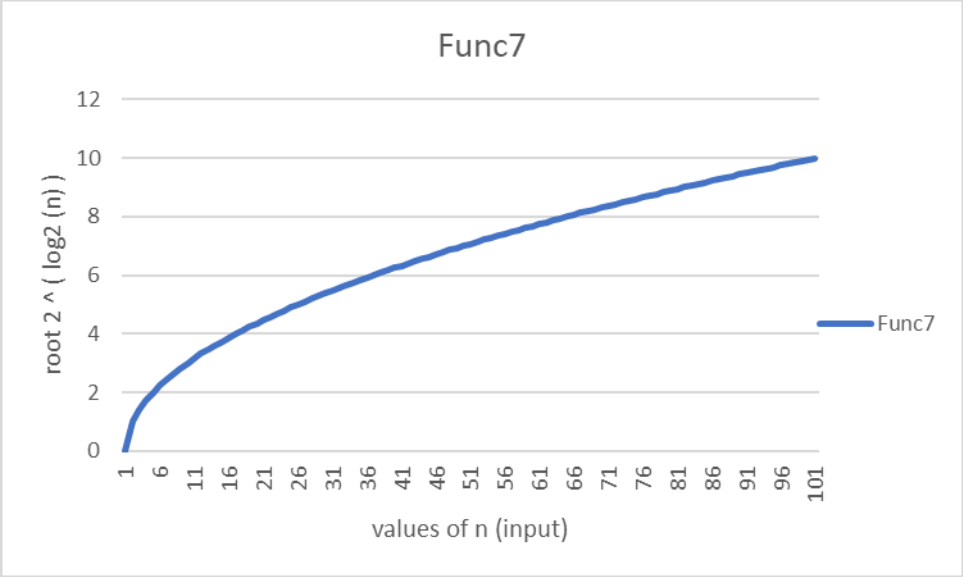
6. Func6= log2(n)

- The graph behaves similar to ln(n). It has infinite slope initially and the slope decreases.



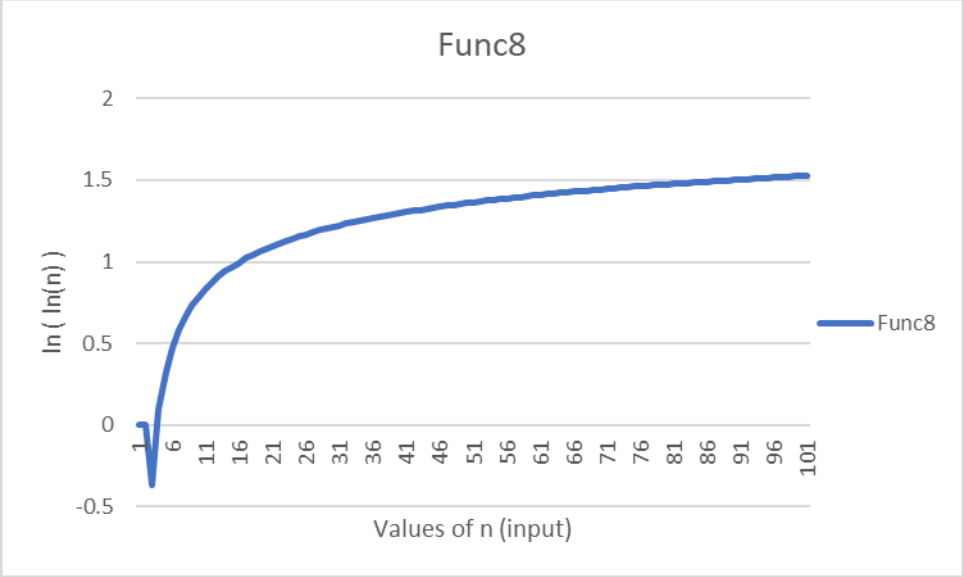
7. Func7= root2 ^ (log2(n))

- The values increases with increasing n, the slope is decreasing gradually as the value of n increases The slope later tends to be constant.



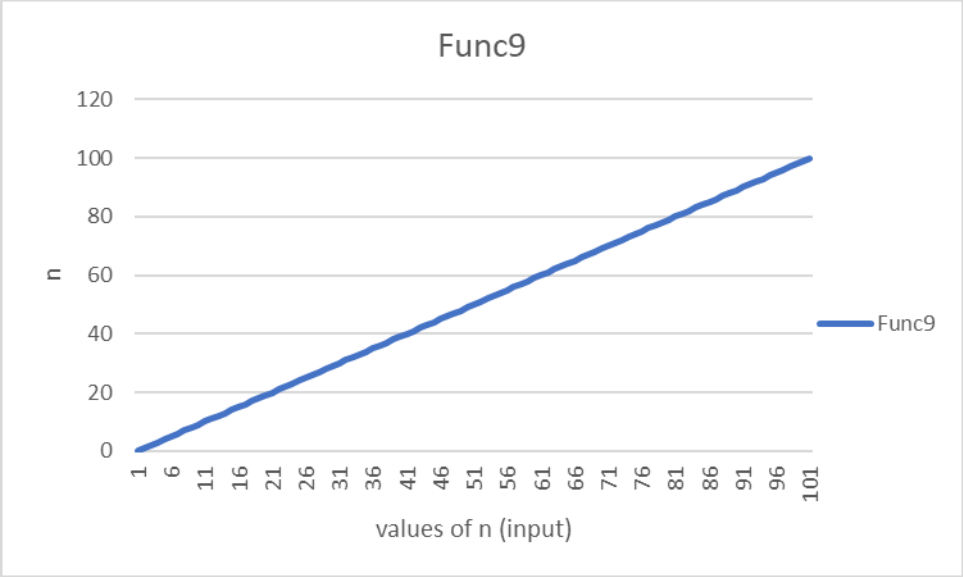
8. Func8= ln(ln(n))

- The values are negative till n=3 and then value of function increases and later tends to be constant from n=80 i.e has very minimum change



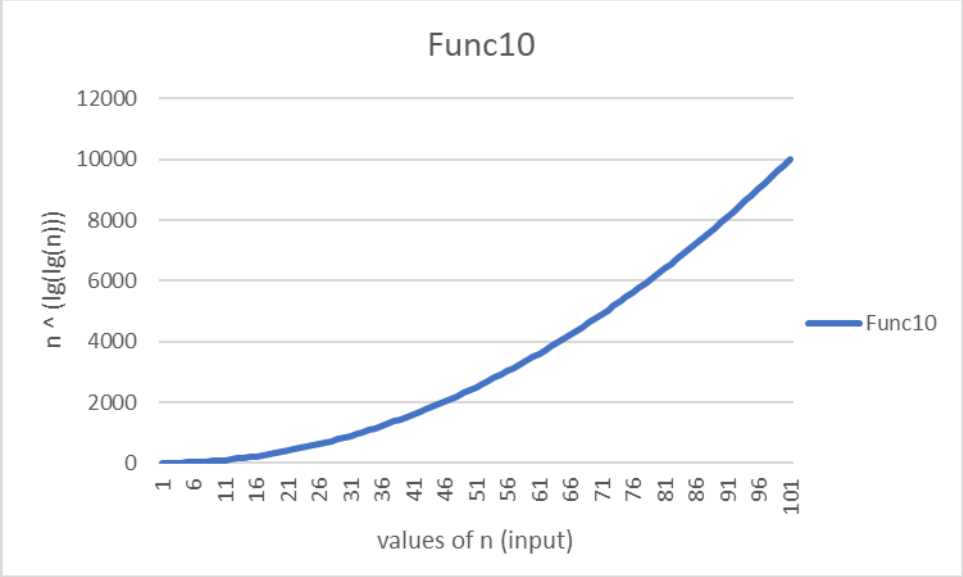
9. Func9= n

- Here the value of function is same as that of the input values. Hence it has a constant positive increasing value slope

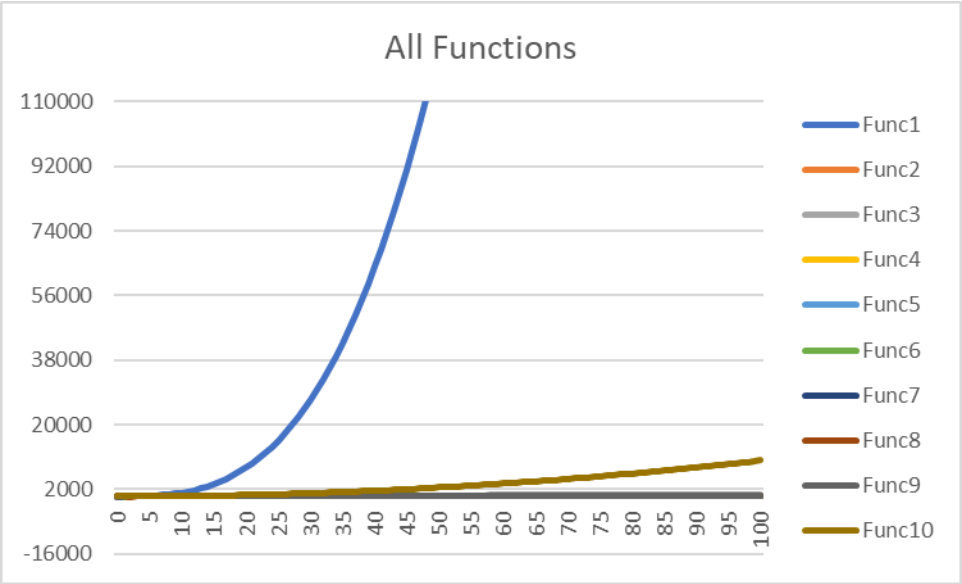


10. Func10= n^ log2(log2(n))

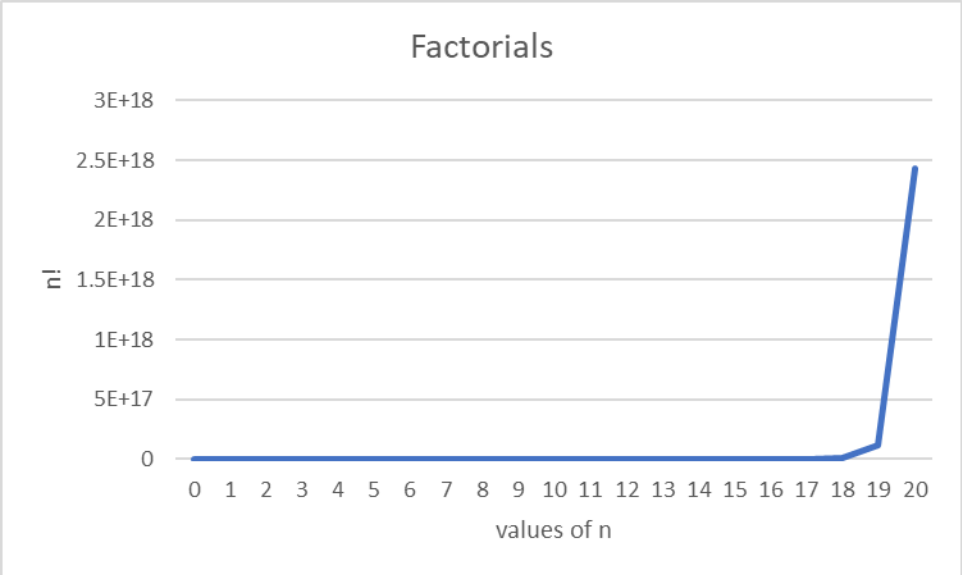
- The value of function increases as the input values increase with an increasing graph and positive slope



RESULT ANALYSIS:



Here I analyze that functions like n^3, n^lg(lg(n)) show similar graph behavior i.e have concave graphs with increasing positive slopes as the input value increases.
Other graphs like ln(n), sqrt(lg(n)), log2(n) show similar nature graphs having decreasing slopes as the value of input increases.
Graph of function n^3 i.e. func1 shows the maximum increase with greater n



When n! is added to the functions, the graph becomes like above, since the values in y axes are very large, the graph appears to be constant till n=18 and then there is sudden increase in the value till n=20

CONCLUSION:	In this experiment, I understood how to implement various functions like linear, non-linear , exponential etc, use the results to form a 2D line graph in microsoft excel and analyze the graphs and its nature for various functions