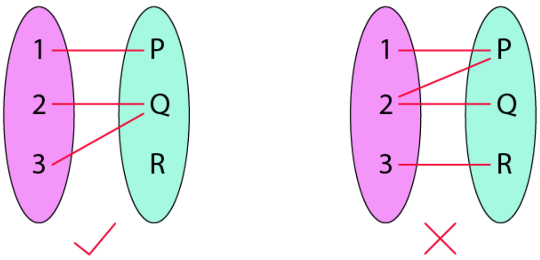
Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.

Experiment No. 0

Aim – To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

Details – 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.



Problem Definition & Assumptions – For this experiment, you have to implement at least 10 functions from the following list.

$(\frac{3}{2})^n$	n^3	$\lg^2 n$	$\lg(n!)$	2^{2^n}	$n^{1/\lg n}$
$\ln \ln n$	$\lg\ n$	$n \cdot 2^n$	$n^{\lg \lg n}$	$\ln n$	$2^{\lg\ n}$
$2^{\lg n}$	$(\lg n)^{\lg n}$	e^n	$(\lg n)!$	$(\sqrt{2})^{\lg n}$	$\sqrt{\lg n}$
$\lg\ (\lg n)$	$2^{\sqrt{2 \lg n}}$	n	2^n	$n \lg n$	$2^{2^{n+1}}$

Note – *lg* denotes for log_2 and *le* denotes log_e

The input (i.e. *n*) to all the above functions varies from 0 to 100 with increment of 1. Then add the function n! in the list and execute the same for n from 0 to 20.

Important Links:

1. C/C++ Function Online library
   https://cplusplus.com/reference/cstdlib/rand/
2. Formal definition of Function
   https://www.whitman.edu/mathematics/higher_math_online/section04.01.html
3. Draw 2-D plot using OpenLibre/MS Excel
   https://support.microsoft.com/en-us/topic/present-your-data-in-a-scatter-chart-or-a-line-chart-4570a80f-599a-4d6b-a155-104a9018b86e

---

Input –

1) Each student randomly chose any ten functions from the aforementioned list.

Output –

1) Print the values of each function value for all *n* starting 0 to 100 in tabular format for both aforementioned cases

2) Draw two 2D plot of all functions such that x-axis represents the values of *n* and y-axis represent the function value for different n values using LibreOffice Calc/MS Excel.

Experiment No.	0
Aim	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.
Name	Allen Andrew
UID No.	2021300006
Class & Division	SE Comps A

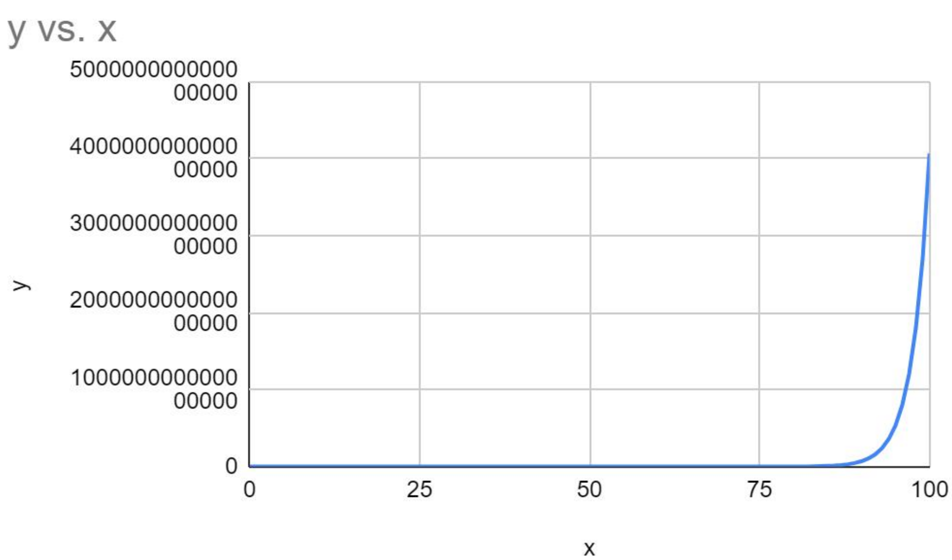
1) $(1.5)^n$

main.cpp

```

#include <iostream>
#include<cmath>

using namespace std;

int main()
{
   int i;
   cout<<"X      Y"<<endl;
   for(i=0;i<=100;i++)
   {
        float y= pow(1.5,i);
        cout<<i<<"      "<<y<<endl;
   }
}

```

X	Y
0	1
1	1.5
2	2.25
3	3.375
4	5.0625
5	7.59375
6	11.3906
7	17.0859
8	25.6289
9	38.4434
10	57.665
11	86.4976
12	129.746
13	194.62
14	291.929
15	437.894
16	656.841
17	985.261
18	1477.89
19	2216.84
20	3325.26
21	4987.89
22	7481.83
23	11222.7
24	16834.1
25	25251.2
26	37876.8
27	56815.1

72	4.77057e+12
73	7.15586e+12
74	1.07338e+13
75	1.61007e+13
76	2.4151e+13
77	3.62265e+13
78	5.43398e+13
79	8.15097e+13
80	1.22265e+14
81	1.83397e+14
82	2.75095e+14
83	4.12643e+14
84	6.18965e+14
85	9.28447e+14
86	1.39267e+15
87	2.08901e+15
88	3.13351e+15
89	4.70026e+15
90	7.05039e+15
91	1.05756e+16
92	1.58634e+16
93	2.37951e+16
94	3.56926e+16
95	5.35389e+16
96	8.03084e+16
97	1.20463e+17
98	1.80694e+17
99	2.71041e+17
100	4.06561e+17



The graph of $y = 1.5^x$ is a exponential function where the base is 1.5. As x increases, y will increase at an increasing rate, approaching infinity. The graph will start at (0,1) and will go upwards and to the right without bound.

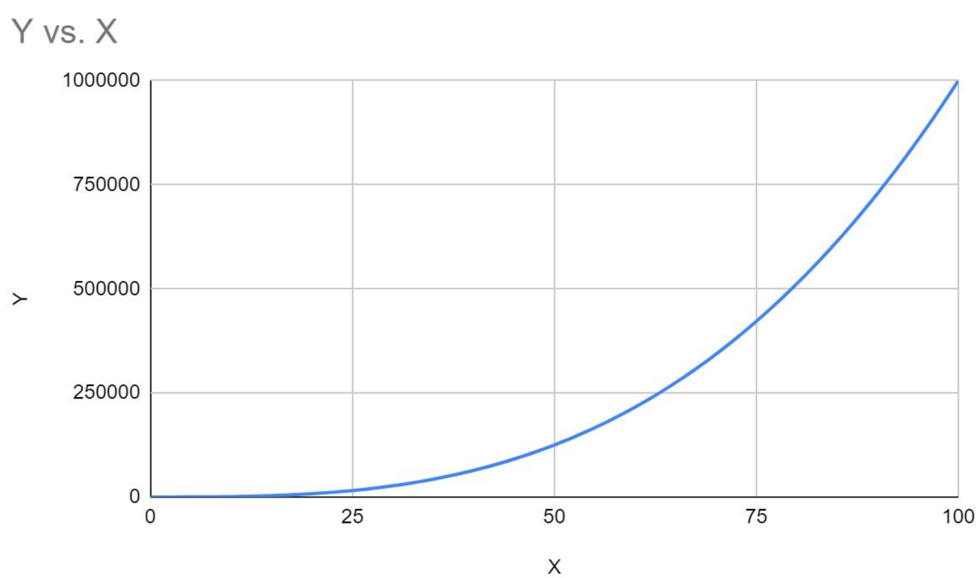
2) n^3

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;

int main()
{
    int i;
    cout<<"X       Y"<<endl;
    for(i=0;i<=100;i++)
    {
        float y= pow(i,3);
        cout<<i<<"      "<<y<<endl;
    }
}


```

X	Y
0	0
1	1
2	8
3	27
4	64
5	125
6	216
7	343
8	512
9	729
10	1000
11	1331
12	1728
13	2197
14	2744
15	3375
16	4096
17	4913
18	5832
19	6859
20	8000
21	9261
22	10648
23	12167
24	13824
25	15625
26	17576
27	19683
28	21952
29	24389
30	27000

69	328509
70	343000
71	357911
72	373248
73	389017
74	405224
75	421875
76	438976
77	456533
78	474552
79	493039
80	512000
81	531441
82	551368
83	571787
84	592704
85	614125
86	636056
87	658503
88	681472
89	704969
90	729000
91	753571
92	778688
93	804357
94	830584
95	857375
96	884736
97	912673
98	941192
99	970299
100	1e+06



In conclusion, the graph of $y=x^3$ is a parabola that opens upwards and has its vertex at the origin (0,0). As x increases or decreases, the value of y also increases or decreases at an increasing rate, respectively. This graph represents a cubic function and can be used to model real-world situations such as the relationship between volume and side length in a cube or the cost of production in a manufacturing process.

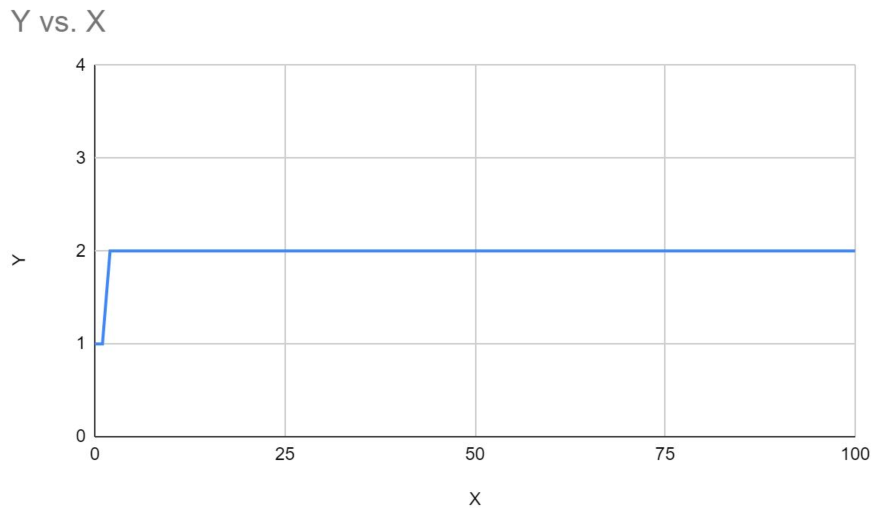
3) n $^{1/lgn}$

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;

int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        float z=log(2)/log(i);
        float y=pow(i,z);
        cout<<i<<"      "<<y<<endl;
    }
}

```

```
X      Y
0      1
1      1
2      2
3      2
4      2
5      2
6      2
7      2
8      2
9      2
10      2
11      2
12      2
13      2
14      2
15      2
16      2
17      2
18      2
19      2
20      2
21      2
22      2
23      2
24      2
25      2
26      2
27      2
28      2
29      2
30      2
```

```
69      2
70      2
71      2
72      2
73      2
74      2
75      2
76      2
77      2
78      2
79      2
80      2
81      2
82      2
83      2
84      2
85      2
86      2
87      2
88      2
89      2
90      2
91      2
92      2
93      2
94      2
95      2
96      2
97      2
98      2
99      2
100      2
```



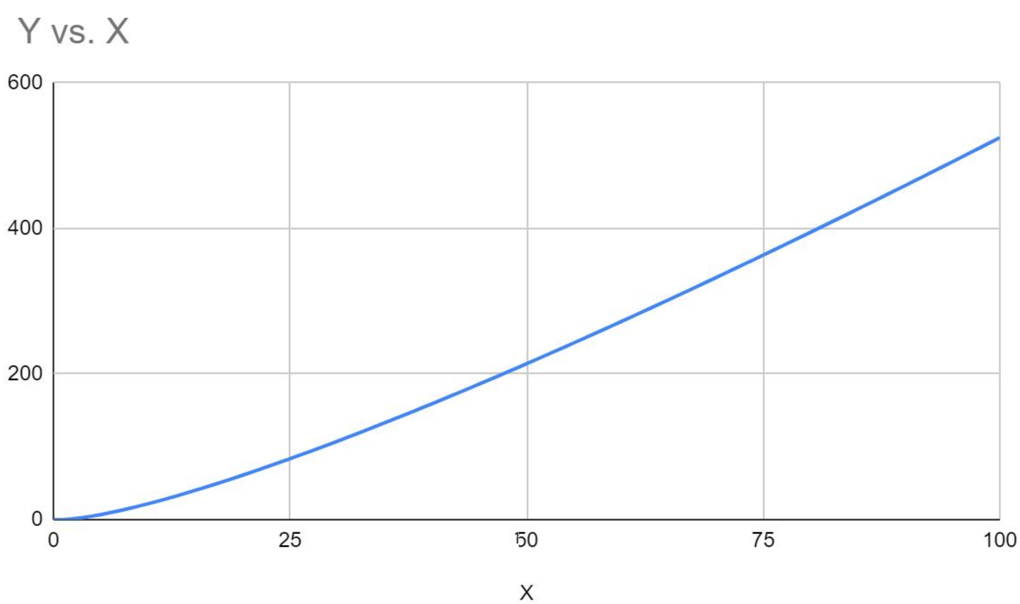
The function is not defined for x=0,1.As x tends to infinity, value of y is a constant i.e. 2.

4) lg(n!)

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;

double factorial(int n)
{
    double f=1.0;
    if(n==0 || n==1)
    return 1;

    for(int i=1;i<=n;i++)
    {
        f=f*i;
    }
    return f;
}

int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        double y=log(factorial(i))/log(2);
        cout<<i<<"      "<<y<<endl;
    }
}
```

X	Y
0	0
1	0
2	1
3	2.58496
4	4.58496
5	6.90689
6	9.49185
7	12.2992
8	15.2992
9	18.4691
10	21.7911
11	25.2505
12	28.8355
13	32.5359
14	36.3432
15	40.2501
16	44.2501
17	48.3376
18	52.5075
19	56.7555
20	61.0774
21	65.4697
22	69.9291
23	74.4527
24	79.0377
25	83.6815
26	88.382
27	93.1368
28	97.9442
29	102.802
30	107.709
70	332.453
71	338.603
72	344.773
73	350.963
74	357.172
75	363.401
76	369.649
77	375.916
78	382.201
79	388.505
80	394.827
81	401.167
82	407.524
83	413.899
84	420.292
85	426.701
86	433.127
87	439.57
88	446.03
89	452.505
90	458.997
91	465.505
92	472.029
93	478.568
94	485.122
95	491.692
96	498.277
97	504.877
98	511.492
99	518.121
100	524.765



The graph of log(n!) represents the logarithm of the factorial of n as a function of n. It is an increasing function that grows very quickly for large values of n. The growth rate of log(n!) can be approximated by Stirling's formula, which states that log(n!) ~ n log(n) - n for large values of n.

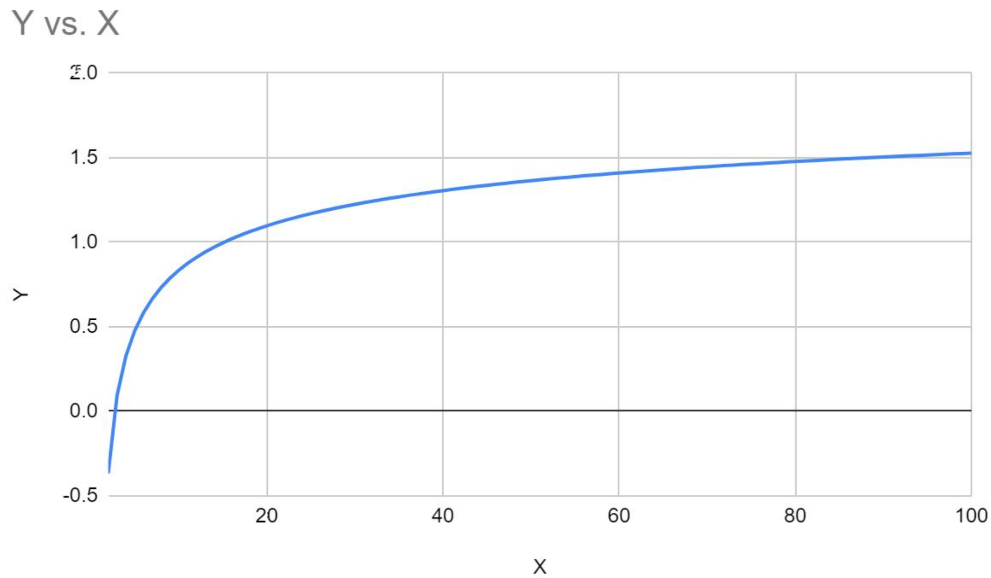
5) ln(ln(n))

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;


int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        double y=log(log(i));
        cout<<i<<"      "<<y<<endl;
    }
}

```

```
X     Y
0     -nan
1     -inf
2     -0.366513
3     0.0940478
4     0.326634
5     0.475885
6     0.583198
7     0.66573
8     0.732099
9     0.787195
10     0.834032
11     0.874591
12     0.910235
13     0.941939
14     0.970422
15     0.996229
16     1.01978
17     1.04141
18     1.06139
19     1.07992
20     1.09719
21     1.11334
22     1.12851
23     1.14279
24     1.15627
25     1.16903
26     1.18114
27     1.19266
28     1.20363
29     1.21411
30     1.22413
```

```
69     1.44317
70     1.44656
71     1.4499
72     1.45317
73     1.45639
74     1.45956
75     1.46267
76     1.46574
77     1.46875
78     1.47172
79     1.47464
80     1.47751
81     1.48034
82     1.48313
83     1.48588
84     1.48858
85     1.49125
86     1.49388
87     1.49647
88     1.49903
89     1.50155
90     1.50404
91     1.50649
92     1.50891
93     1.5113
94     1.51365
95     1.51598
96     1.51828
97     1.52054
98     1.52278
99     1.52499
100     1.52718
```



ln(ln(x)) is an increasing function for x > e (where e is the mathematical constant approximately equal to 2.71828). This means that as x increases, the value of ln(ln(x)) will also increase. For x < e, ln(ln(x)) is a decreasing function, so as x decreases, the value of ln(ln(x)) will also decrease.

ln(ln(x)) has some important properties. For example, as x approaches infinity, ln(ln(x)) approaches infinity, which means that for extremely large values of x, the value of ln(ln(x)) will become very large

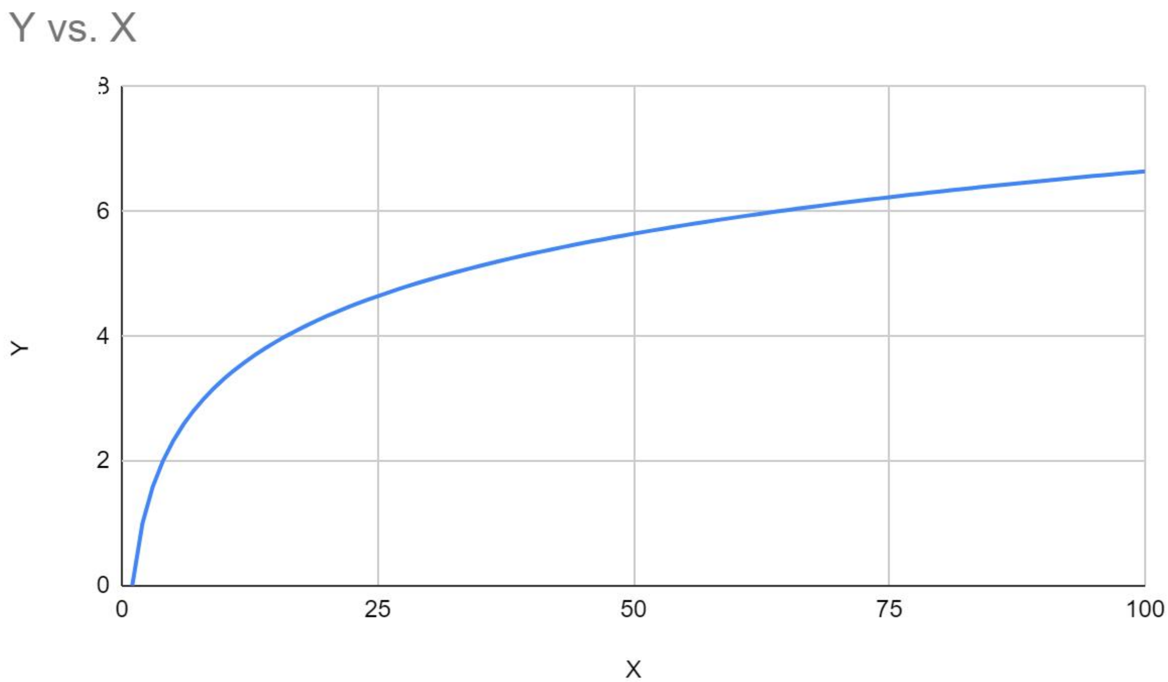
6) lg(n)

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;


int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        double y=log(i)/log(2);
        cout<<i<<"      "<<y<<endl;
    }
}

```

```
X      Y
0      -inf
1      0
2      1
3      1.58496
4      2
5      2.32193
6      2.58496
7      2.80735
8      3
9      3.16993
10      3.32193
11      3.45943
12      3.58496
13      3.70044
14      3.80735
15      3.90689
16      4
17      4.08746
18      4.16993
19      4.24793
20      4.32193
21      4.39232
22      4.45943
23      4.52356
24      4.58496
25      4.64386
26      4.70044
27      4.75489
28      4.80735
29      4.85798
30      4.90689
```

```
70      6.12928
71      6.14975
72      6.16993
73      6.18982
74      6.20945
75      6.22882
76      6.24793
77      6.26679
78      6.2854
79      6.30378
80      6.32193
81      6.33985
82      6.35755
83      6.37504
84      6.39232
85      6.40939
86      6.42626
87      6.44294
88      6.45943
89      6.47573
90      6.49185
91      6.50779
92      6.52356
93      6.53916
94      6.55459
95      6.56986
96      6.58496
97      6.59991
98      6.61471
99      6.62936
100      6.64386
```

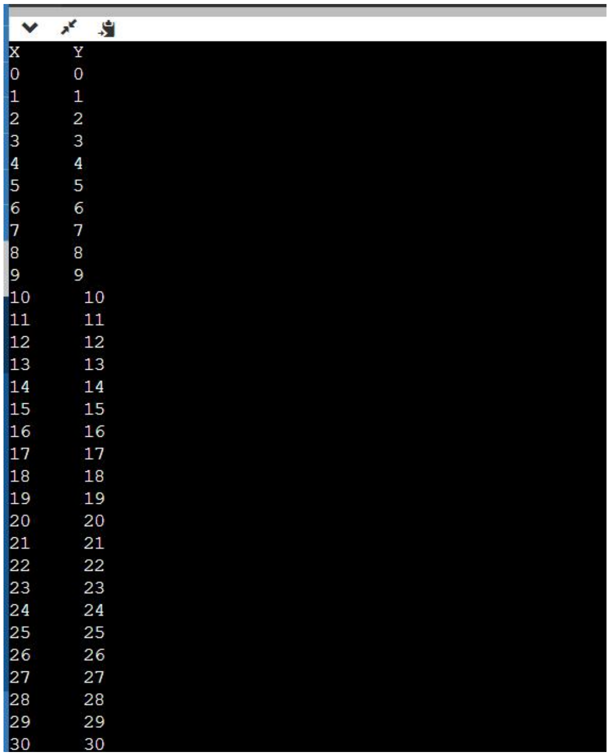


The graph of this function roughly imitates the one belonging to a generic logarithmic function having a base value greater than 1.
This function is undefined at the input value of 0, the output at which is shown as 0 in the graph as per the default behaviour of the graph in an excel file.
This function, being a logarithmic one, succeeds in providing proper outputs even when the inputs are in millions or billions.

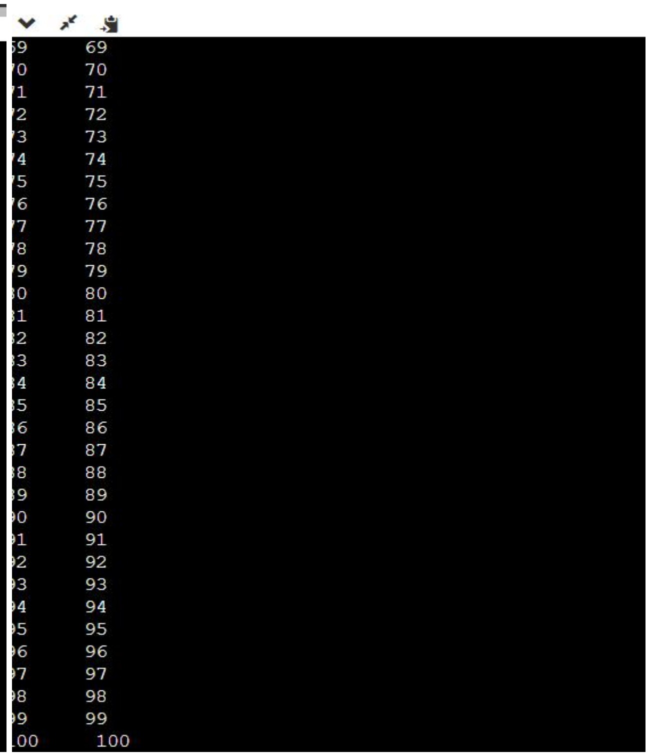
7) 2^{lgn}

main.cpp

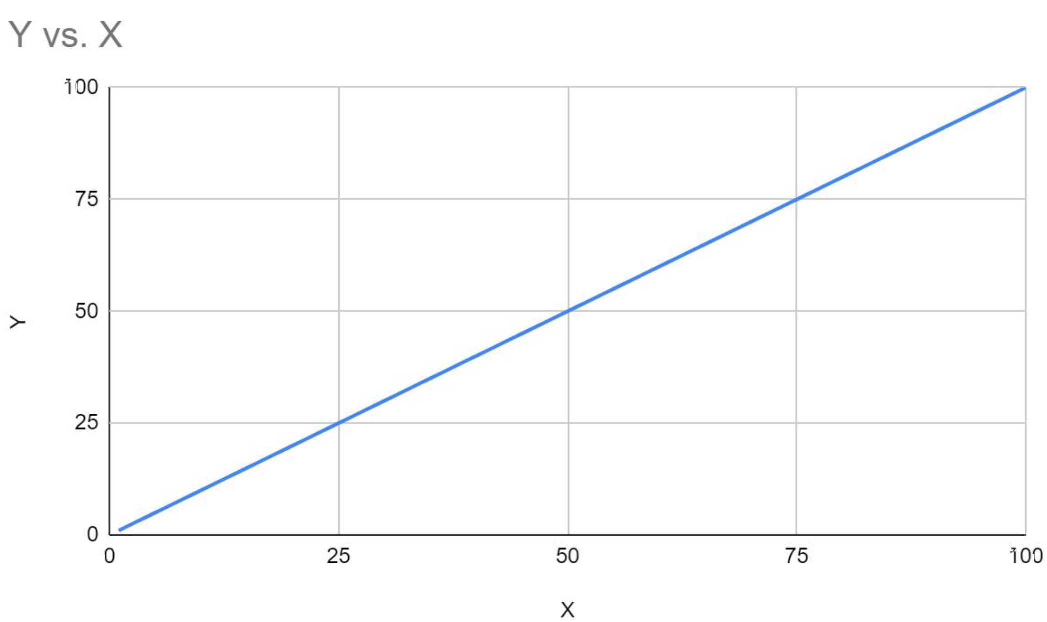
```
#include <iostream>
#include<cmath>

using namespace std;


int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        double z=log(i)/log(2);
        double y=pow(2,z);
        cout<<i<<"      "<<y<<endl;
    }
}

```



X	Y
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30



69	69
70	70
71	71
72	72
73	73
74	74
75	75
76	76
77	77
78	78
79	79
80	80
81	81
82	82
83	83
84	84
85	85
86	86
87	87
88	88
89	89
90	90
91	91
92	92
93	93
94	94
95	95
96	96
97	97
98	98
99	99
100	100



The graph is not defined for x=0. The graph shows a linear behavior for other values of x.

This behavior could be predicted by basic log properties.

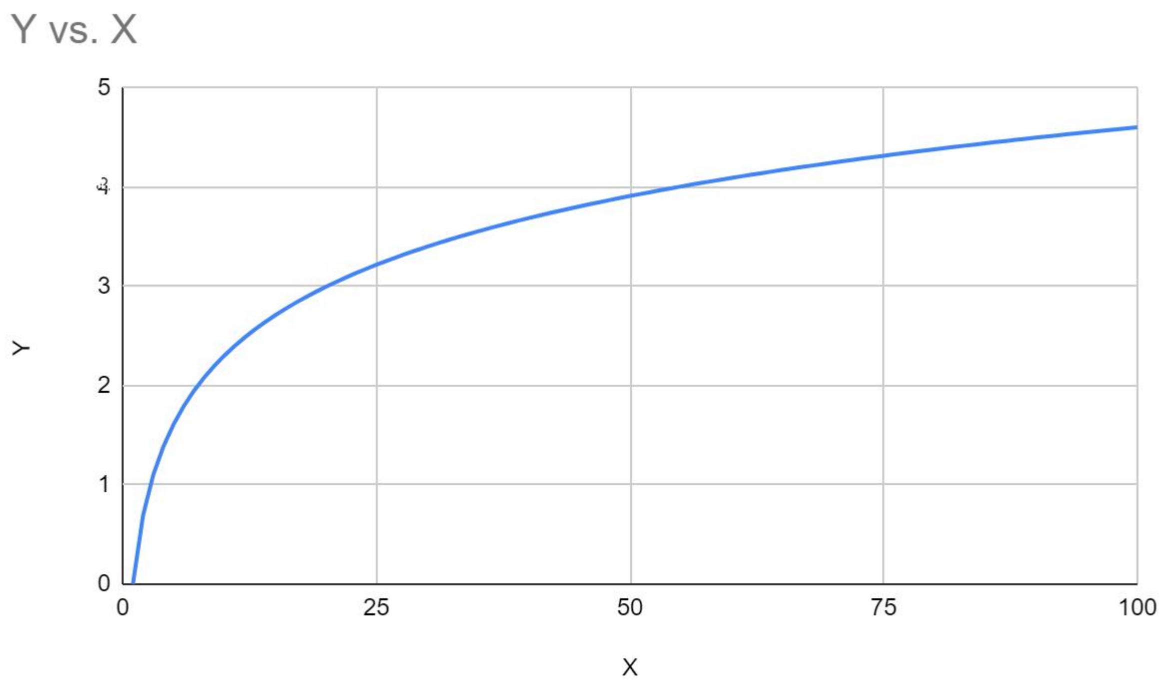
8) ln(x)

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;


int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        double y=log(i);
        cout<<i<<"       "<<y<<endl;
    }
}
```

```
X      Y
0      -inf
1      0
2      0.693147
3      1.09861
4      1.38629
5      1.60944
6      1.79176
7      1.94591
8      2.07944
9      2.19722
10      2.30259
11      2.3979
12      2.48491
13      2.56495
14      2.63906
15      2.70805
16      2.77259
17      2.83321
18      2.89037
19      2.94444
20      2.99573
21      3.04452
22      3.09104
23      3.13549
24      3.17805
25      3.21888
26      3.2581
27      3.29584
28      3.3322
29      3.3673
30      3.4012
```

```
69      4.23411
70      4.2485
71      4.26268
72      4.27667
73      4.29046
74      4.30407
75      4.31749
76      4.33073
77      4.34381
78      4.35671
79      4.36945
80      4.38203
81      4.39445
82      4.40672
83      4.41884
84      4.43082
85      4.44265
86      4.45435
87      4.46591
88      4.47734
89      4.48864
90      4.49981
91      4.51086
92      4.52179
93      4.5326
94      4.54329
95      4.55388
96      4.56435
97      4.57471
98      4.58497
99      4.59512
100      4.60517
```



The graph of this function roughly imitates the one belonging to a generic logarithmic function having a base value greater than 1.
This function is undefined at the input value of 0, the output at which is shown as 0 in the graph as per the default behaviour of the graph in an excel file.
This function, being a logarithmic one, succeeds in providing proper outputs even when the inputs are in millions or billions.

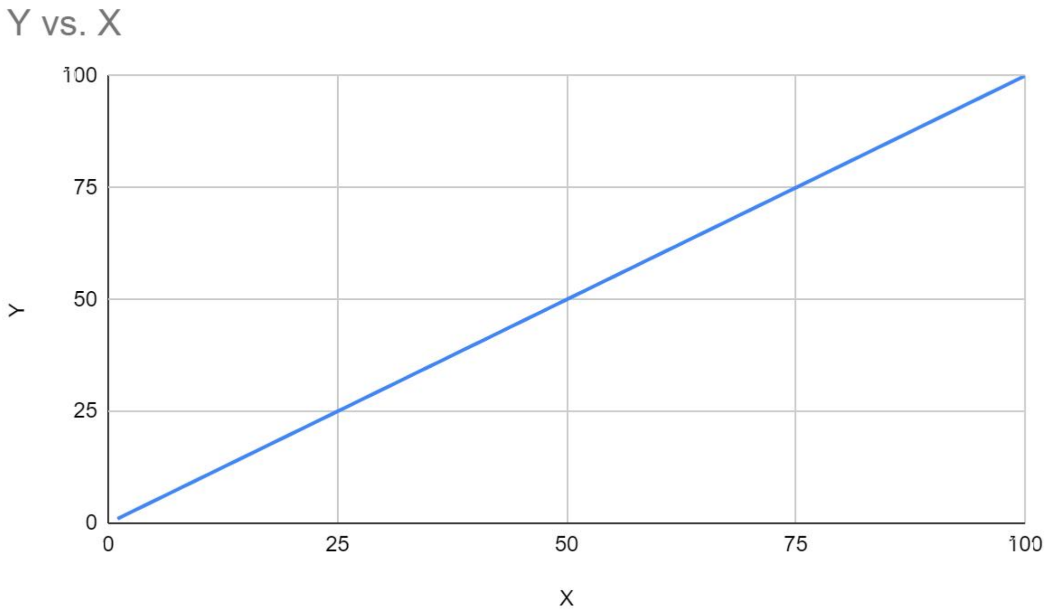
9) n

main.cpp

```cpp
#include <iostream>
#include<cmath>

using namespace std;


int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        int y=i;
        cout<<i<<"      "<<y<<endl;
    }
}
```

```
X      Y
0      0
1      1
2      2
3      3
4      4
5      5
6      6
7      7
8      8
9      9
10      10
11      11
12      12
13      13
14      14
15      15
16      16
17      17
18      18
19      19
20      20
21      21
22      22
23      23
24      24
25      25
26      26
27      27
28      28
29      29
30      30
```

```
69      69
70      70
71      71
72      72
73      73
74      74
75      75
76      76
77      77
78      78
79      79
80      80
81      81
82      82
83      83
84      84
85      85
86      86
87      87
88      88
89      89
90      90
91      91
92      92
93      93
94      94
95      95
96      96
97      97
98      98
99      99
100      100
```



The graph is linear in nature.Value of y is equal to x for every x . Domain of function is R and so is Range.

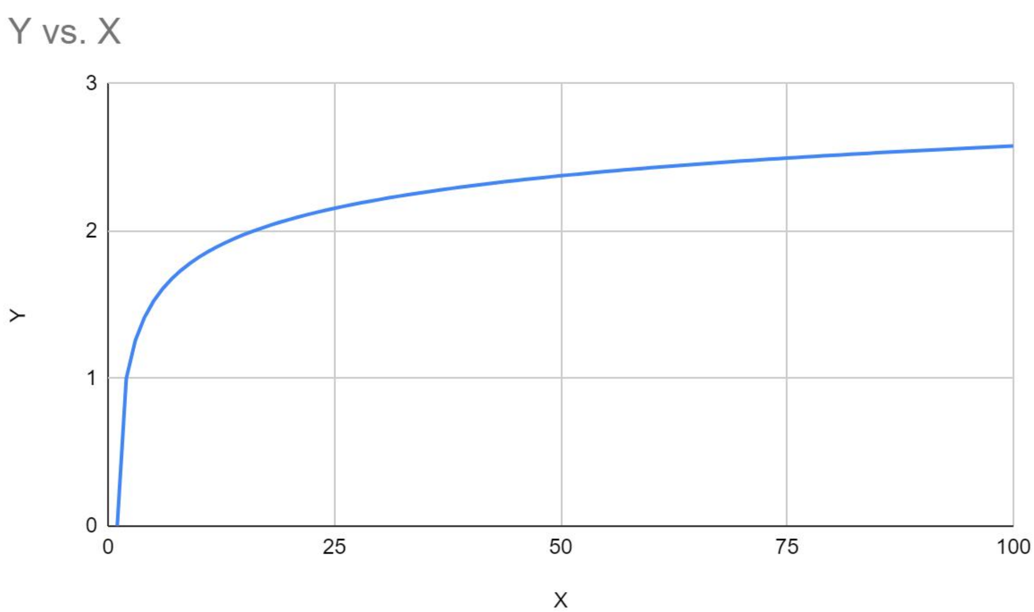
10) $(\lg n)^{1/2}$

main.cpp

```
#include <iostream>
#include<cmath>

using namespace std;


int main()
{
    int i;
    cout<<"X      Y"<<endl;
    for(i=0;i<=100;i++)
    {
        double z=log(i)/log(2);
        double y=sqrt(z);
        cout<<i<<"      "<<y<<endl;
    }
}
```

0	-nan
1	0
2	1
3	1.25895
4	1.41421
5	1.52379
6	1.60778
7	1.67552
8	1.73205
9	1.78043
10	1.82262
11	1.85995
12	1.8934
13	1.92365
14	1.95124
15	1.97659
16	2
17	2.02175
18	2.04204
19	2.06105
20	2.07892
21	2.09579
22	2.11174
23	2.12687
24	2.14125
25	2.15496
26	2.16805
27	2.18057
28	2.19257
29	2.20408
30	2.21515
31	2.2258

69	2.47154
70	2.47574
71	2.47987
72	2.48393
73	2.48794
74	2.49188
75	2.49576
76	2.49959
77	2.50336
78	2.50707
79	2.51073
80	2.51434
81	2.51791
82	2.52142
83	2.52488
84	2.5283
85	2.53168
86	2.53501
87	2.5383
88	2.54154
89	2.54475
90	2.54791
91	2.55104
92	2.55413
93	2.55718
94	2.56019
95	2.56317
96	2.56612
97	2.56903
98	2.57191
99	2.57475
100	2.57757



√log(x) is an increasing function: the square root of logarithm is an increasing function, which means that as x increases, $\sqrt{\log(x)}$ also increases.

Conclusion:

By performing this experiment, I was able to observe the difference in the various functions that were implemented. I was also able to understand the procedure of plotting a graph from the obtained data using Microsoft excel.