

|  |  |
| --- | --- |
| Name | Vedant Deshmukh |
| UID No. | 2021300025 |
| Class | COMPS A (B batch) |
| Experiment No. | 01 A |

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

Algorithm:

1. *f(n) = n*

Algorithm:

F(n): Return n

1. *f(n) = log(n)*

Algorithm:

F(n): Return log2(n)

1. *f(n) = log (log n)*

Algorithm:

F(n) : Return log2(log2(n))



1. *f(n) = n log n*

Algorithm:

F(n) :

if n = 0

Error

else return n \* log2(n)

1. *f(n) = (log n)log(n)*

Algorithm:

F(n):

If n = 0 Error Else Return pow(log (n), log (n))

***6)*** *f(n) =* √𝑙𝑜𝑔2(𝑛)

Algorithm:

F(n): If n = 0 Error else Return sqrt(log2(n))

1. *f(n) = 2log(n)*

Algorithm:

F(n): If n = 0 Error else Return pow(2, log2(n))

1. *f(n) = (log n)!*

LogFactorial(n): If n <= 0

Error // as log of 0 does not exist

Else



fact = 1

for i from 1 to ceil(log n) do fact = fact \* i

Return fact

***9)*** *f(n) = (*√*2)log n*

Algorithm:

F(n): If n = 0 Error else Return pow(sqrt(2), log n)

***10)*** *f(n) = n3*

Algorithm:

F(n): Return n \* n \* n

Code:



nclude <math.h> nclude <std o.h>

nt fn\_01( nt n) { return n;

}

double fn\_02( nt n) {

f (n <= 0) return -1.0; return log2(n);

}

double fn\_03( nt n) {

f (n <= 0) return -1.0;

f (log2(n) == 0) return -1.0; return log2(log2(n));

}

double fn\_04( nt n) {

f (n <= 0) return -1.0; return n \* log2(n);

}



double fn\_05( nt n) {



f (n <= 0) return -1.0; return pow(log2(n), log2(n));

}

double fn\_06( nt n) {

f (n <= 0) return -1.0; return sqrt(log2(n));

}

double fn\_07( nt n) {

f (n <= 0) return -1.0; return pow(2, log2(n));

}

// (lg n)!

double fn\_08( nt n) {

f (n <= 0) return -1.0; double fact = 1;

for ( nt = 1; <= ce l(log2(n)); ++) { fact \*= ;

}

return fact;

}

double fn\_09( nt n) {

f (n <= 0) return -1.0;

return pow(sqrt(2), log2(n));

}

double fn\_10( nt n) { return n \* n \* n;

}

double fact( nt n) {

f (n == 0) return 1; double f = 1.0;

for ( nt = 1; <= n; ++) { f = f \* ;

}

return f;

}

nt ma n() { FILE \*fptr;

// fptr = fopen("D://Tejas//College//sem4//daa//data.csv", "w"); fptr = fopen("../csv/funct ons.csv", "w");





fpr ntf(fptr, " ,n,lg n,lg(lg n),n\*lg n,(lg n)^(lg n),sqrt(lg n),2^lg n,(lg n)!,sqrt(2)^lg n,n^3,n!\n");

pr ntf("\n | n lg n lg(lg n) n\*lg n (lg n)^(lg n) sqrt(lg n) 2^lg n (lg n)! sqrt(2)^lg n n^3\n\n");

for ( nt = 0; <= 100; ++) {

pr ntf("%3d | %3d %5.2f %8.2f %8.2f %10.2f %10.2f %6.0f %7.0f

%8.2f %10.0f\n", , fn\_01( ), fn\_02( ), fn\_03( ), fn\_04( ), fn\_05( ),

fn\_06( ), fn\_07( ), fn\_08( ), fn\_09( ), fn\_10( ));

fpr ntf(fptr, "%d,%d,%f,%f,%f,%f,%f,%.0f,%.0f,%f,%.0f", , fn\_01( ),

fn\_02( ), fn\_03( ), fn\_04( ), fn\_05( ), fn\_06( ), fn\_07( ), fn\_08( ), fn\_09( ), fn\_10( ));

f ( <= 20) {

fpr ntf(fptr, ",%0.f\n", fact( ));

} else {

fpr ntf(fptr, ",-\n");

}

}

fclose(fptr);

pr ntf("\n\nFactor als: \n"); for ( nt = 0; <= 20; ++) {

pr ntf("%2d! = %.0f\n", , fact( ));

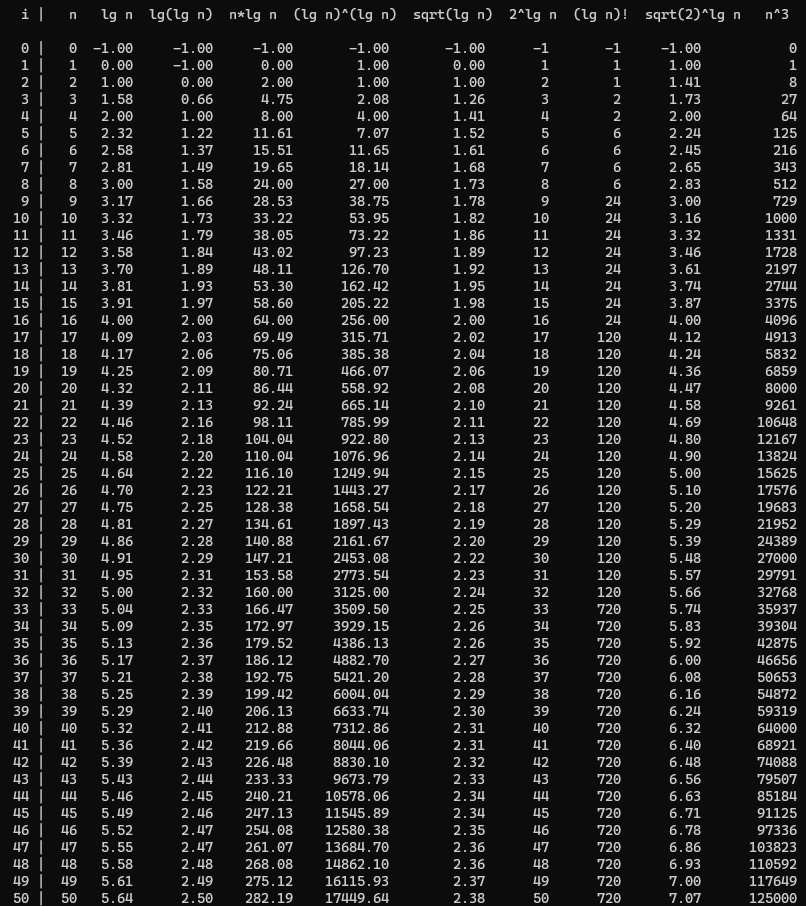
}

return 0;

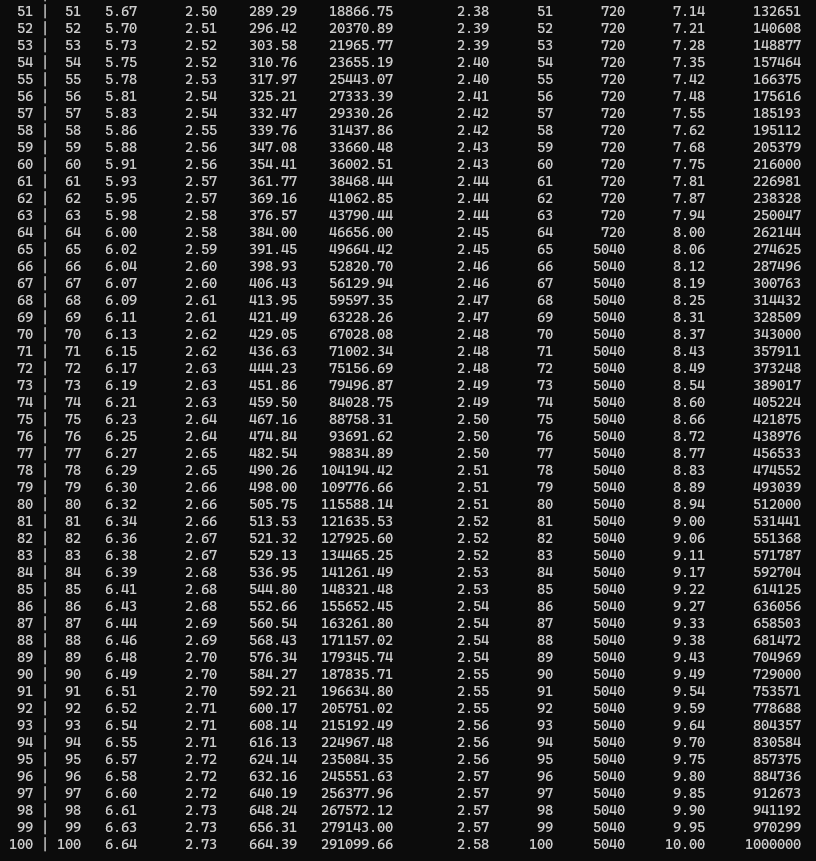
}



Output:









1

5

9

13

17

21

25

29

33

37

41

45

49

53

57

61

65

69

73

77

81

85

89

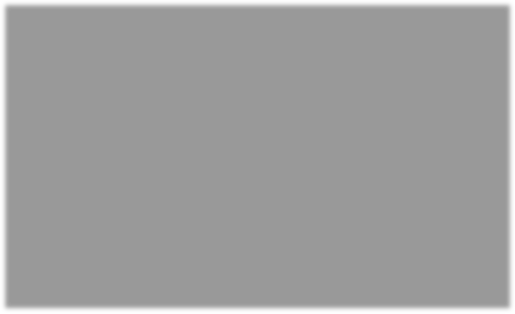
93

97

101



# Chart:



Multiple functions

100

80

60

40

20

0

n

lg n

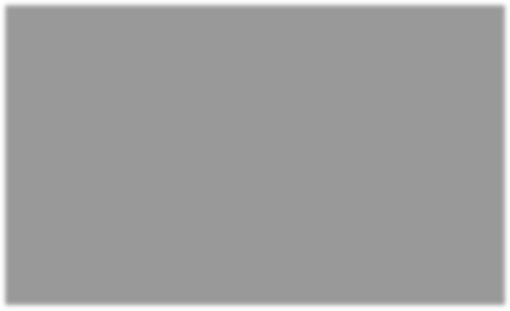
lg(lg n)

n\*lg n

(lg n)^(lg n) sqrt(lg n) 2^lg n (lg n)!

sqrt(2)^lg n n^3 n!





Multiple functions

Large Scale Y axis

10000

8000

6000

4000

2000

0

n

(lg n)^(lg n) sqrt(2)^lg n

lg n

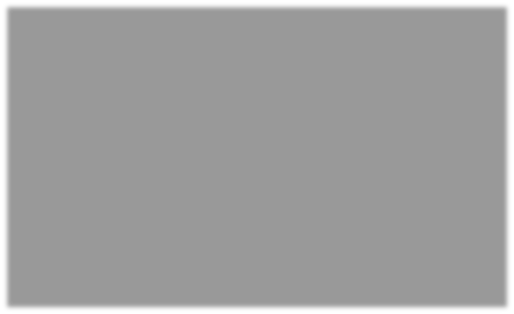
sqrt(lg n) n^3

lg(lg n)

2^lg n n!

n\*lg n

(lg n)!



Multiple Functions

Small Scale Y axis

10

8

6

4

2

0

n

lg n

lg(lg n)

n\*lg n

(lg n)^(lg n) sqrt(lg n) 2^lg n (lg n)!

sqrt(2)^lg n n^3 n!

1

5

9

13

17

21

25

29

33

37

41

45

49

53

57

61

65

69

73

77

81

85

89

93

97

101

1

5

9

13

17

21

25

29

33

37

41

45

49

53

57

61

65

69

73

77

81

85

89

93

97

101



Observations:

|  |  |  |
| --- | --- | --- |
| **Sr** | **Function *f(n)*** | **Observation** |
| 1 | *n* | 1. Linear graph 2. Value increases equally with the input *n* 3. Slope of the graph is 1 4. Steady increase in value |
| 2 | *log(n)* | 1. Logarithmic graph 2. Value increases but slowly 3. Slope of graph is 1 meaning, for greater value of   𝑥 log(2)  n the slope becomes smaller and smaller   1. The function is undefined for 0 |
| 3 | *log (log n)* | 1. Logarithmic graph 2. Value increases even slower compared to *log n* 3. Slope is 1 meaning, for greater values of n,   𝑥2 (𝑙𝑜𝑔 𝑛)2  the slope even smaller compared to *log n*.   1. The function is undefined for n=0 and n=1 |
| 4 | *n log n* | 1. Not exactly linear but not exponential 2. Value keeps on increasing faster compared to *n* 3. Slope is *log n +* 1, implying that it keeps on increasing 4. Undefined at n = 0 |
| 5 | *(log n)log(n)* | 1. Exponential graph 2. Initially, the rate of increase is low as compared to *n log n,* but eventually, it becomes larger and the function completely explodes 3. The slope keeps on increasing 4. Undefined at n = 0 |
| 6 | √log(𝑛) | 1. Logarithmic graph 2. As slow as *log (log n)* 3. Slope becomes smaller and smaller with increasing values of n 4. Undefined at n = 0 |



|  |  |  |
| --- | --- | --- |
| 7 | *2log(n)* | 1. From the property of logarithms, 2log(n) = n 2. Hence this function is essentially the same as *f(n) = n* 3. Therefore, we see a linear graph with slope 1 4. Completely coincides with *f(n) = n* |
| 8 | *(log n)!* | 1. Factorial can only be taken of whole numbers, hence only the ceil of log n is considered 2. We get the same ceil value for a range of log n values. Therefore, we observe a step like behavior in graph 3. The slope is 0 for sometime, then suddenly when ceil(log n) jumps to its next value, the slope becomes infinite and graph jumps on the next level |
| 9 | *(*√*2)log n* | 1. Its growth is slightly greater than log n 2. Grows really slowly for an exponential function 3. Slope slowly keeps on increasing 4. Not defined on n = 0 |
| 10 | *n3* | 1. Cubic graph 2. Grows extremely fast 3. Slope is *3n2* |
| 11 | *n!* | 1. Grows the fastest among the above 10 functions 2. Goes out of bounds immediately 3. Slope grows extremely fast |

# Conclusion:

Through this experiment, I have understood how different functions grow with growing input. I have observed their graph and comparatively studied it. I have understood how the composition of function affects the overall output and how functions compare to each other.