Name	Rajas Baadkar
UID No.	2021300007
Subject	Design And Analysis Of Algorithm
Class	Comps A
Experiment No.	1
AIM	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.

Theory –

In this experiment, we implemented 10 different functions as follows:

1. N cube
2. 2 raised to root(2 log n)
3. (3/2) raised to n
4. n log n
5. ln n
6. 2 raised to 2 raised to n
7. N
8. 2 raised to 2 raised to n+1
9. e raised to n
10. 2 raised to log n

Algorithm –

The algorithm being used here is basically the function to be printed with n ranging from 0 to 100 for each function.

Program –

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

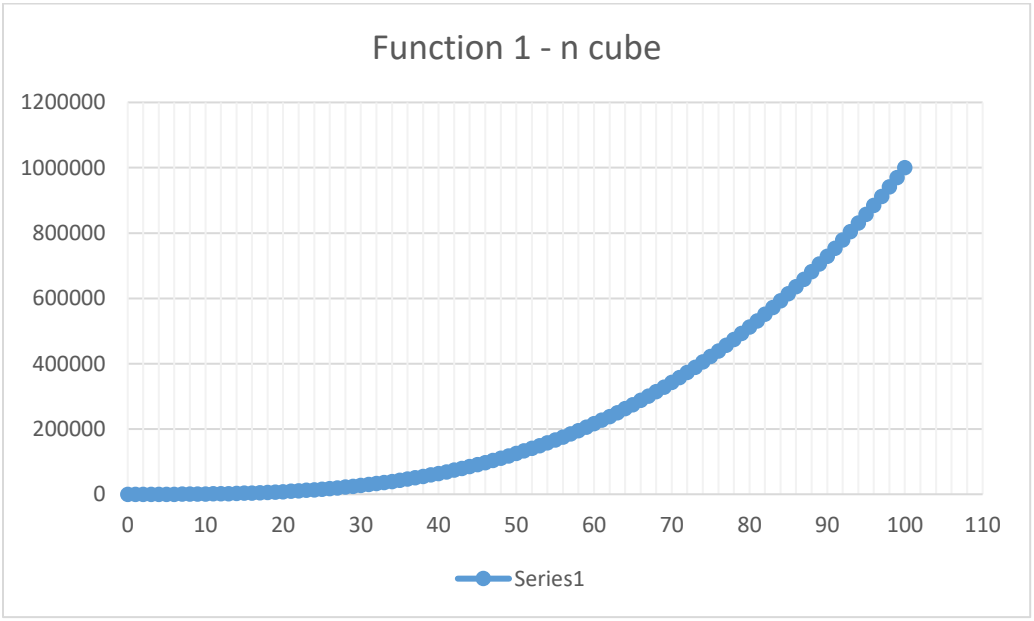
SE – COMP (SE-A) **Sub- DAA Lab**

```c
#include <math.h>

long fact(int num){
    if(num == 0){
        return num;
    }
    else{
        return num*fact(num-1);
    }
}
void main(){
    printf("Function 1: n cube\n");
    for(double i=0;i<=100;i++){
        printf("%.0lf\n",pow(i,3));

    }
    printf("Function 2: 2 raised to root 2 log n\n");
    for(double i = 0;i<=100;i++){
        printf("%.3lf\n", pow(2, sqrt(2*log2(i))));
    }
    printf("Function 3: (3/2) raised to n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",pow((3.0/2.0),i));
    }
    printf("Function 4: n log n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",i * log2(i));
    }
    printf("Function 5: ln n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",log(i));
    }
    printf("Function 6: 2 raised to 2 raised to n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",pow(2,pow(2,i)));
    }
    printf("Function 7: n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",i);
    }
    printf("Function 8: 2 raised to 2 raised to n+1\n");
```

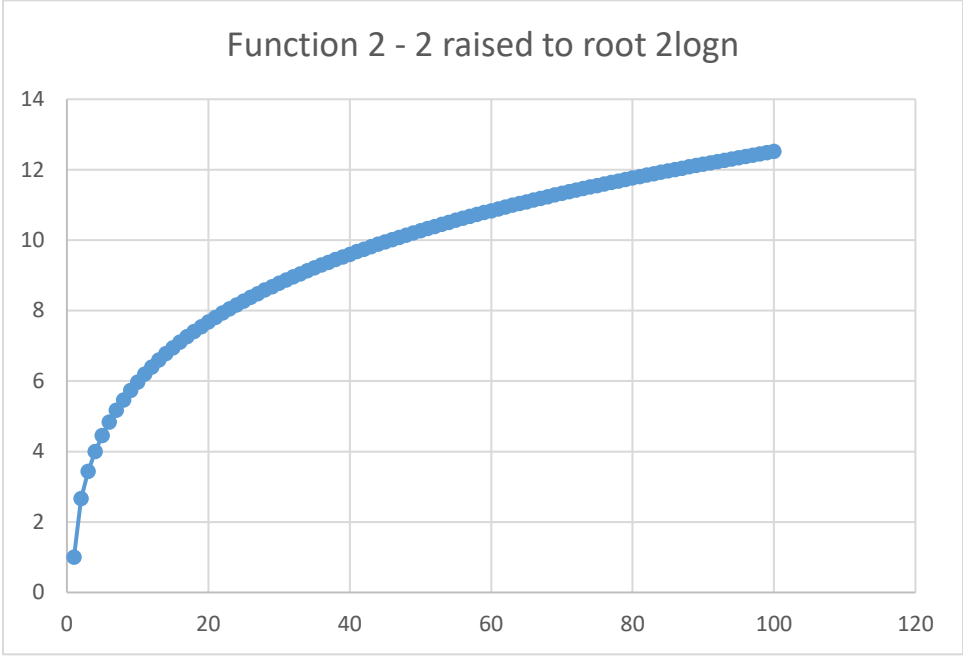
```
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",pow(2,pow(2,i+1)));
    }
    printf("Function 9: e raised to n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",exp(i));
    }
    printf("Function 10: 2 raised to log n\n");
    for(double i = 0;i<=100;i++) {
        printf("%.3lf\n",pow(2,log2(i)));
    }

}
```

Result Analysis –

1.

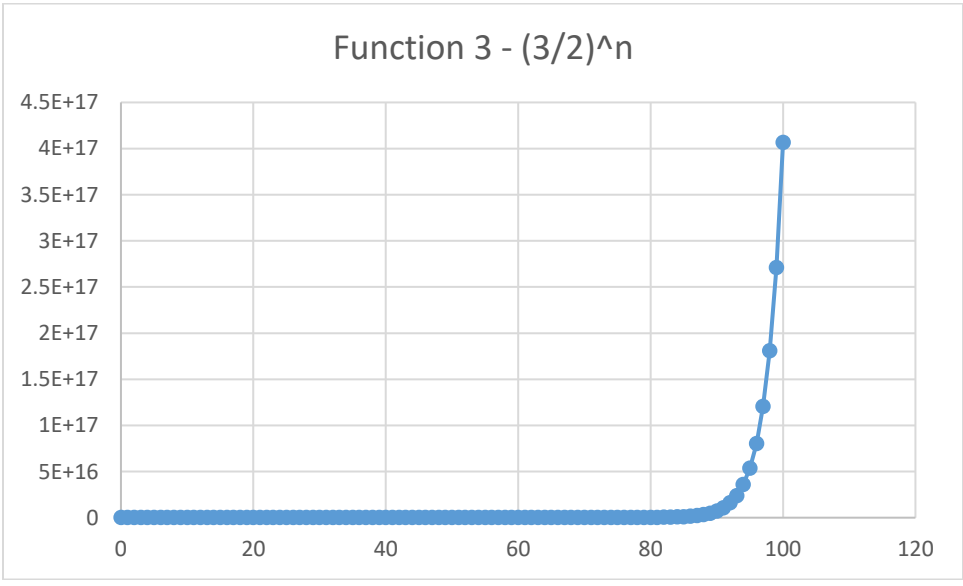


2.



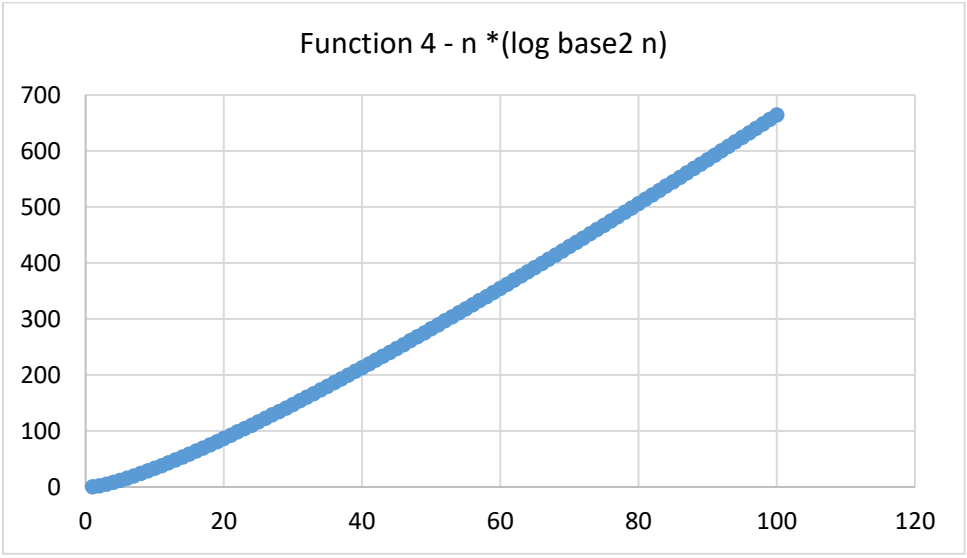
- At n = 0, log 0 does not exist, hence the graph does not exist.

3.



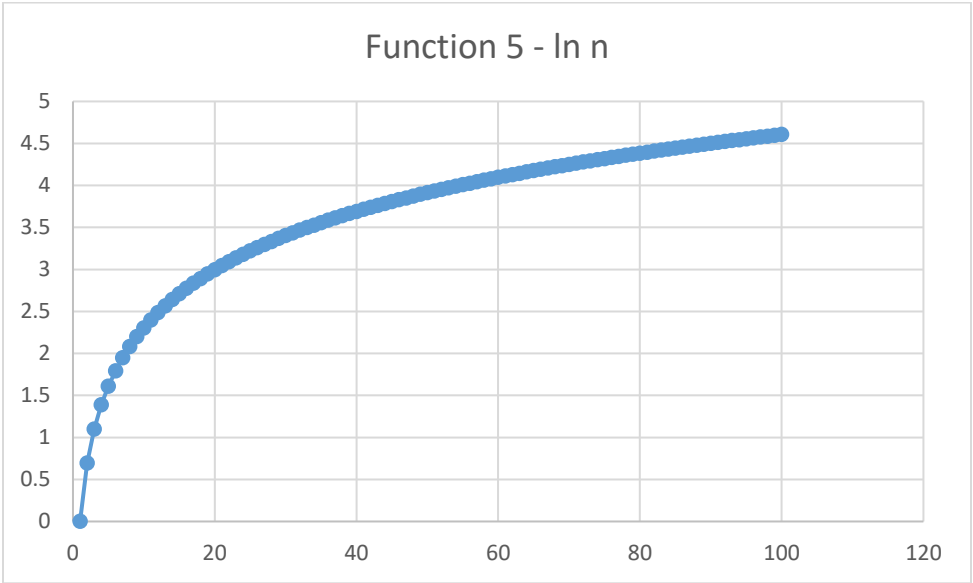
- The graph takes the spike at point n = 87.

4.



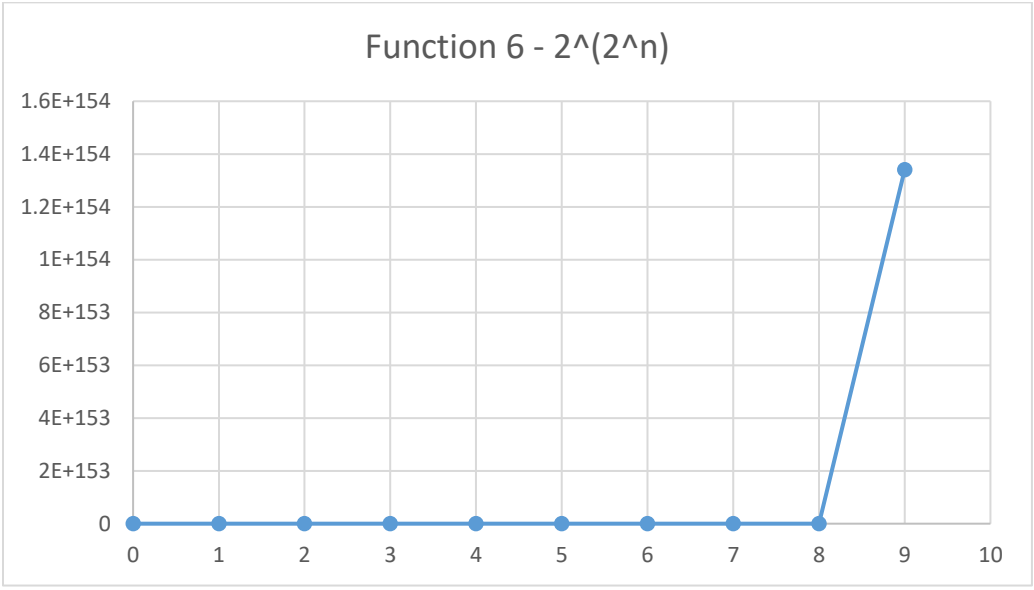
- log 0 does not exist, hence at n = 0 the graph does not exist.

5.



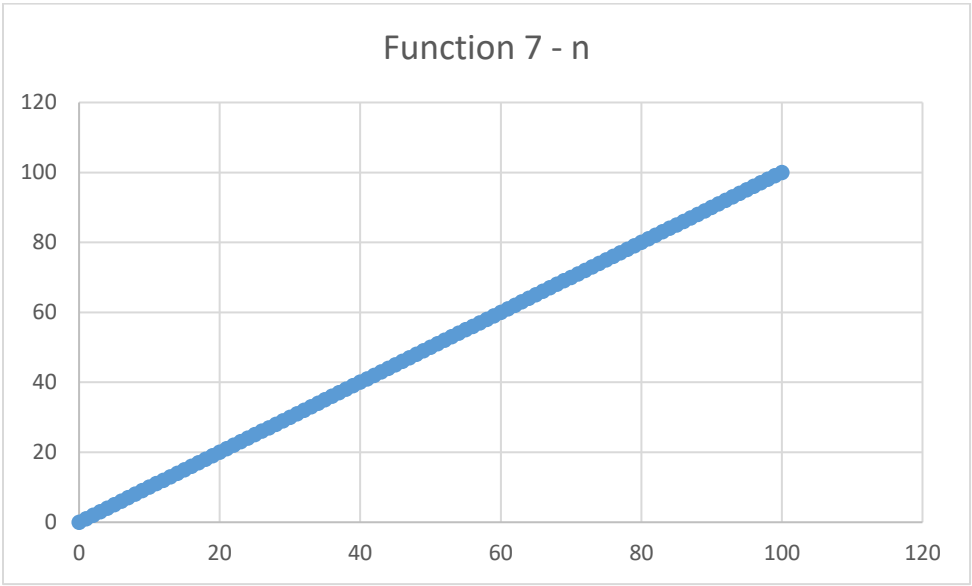
- Ln 0 does not exist, hence at n = 0 the graph does not exist.

6.



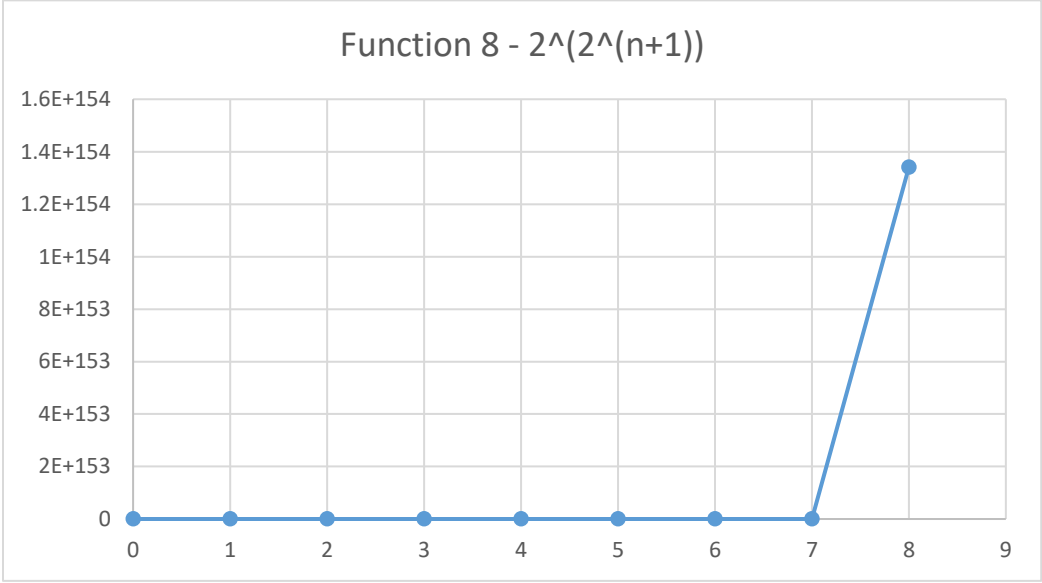
- Here the graph is existing upto n = 9, after that the compiler is not able to compute values as it goes out of bound.

7.



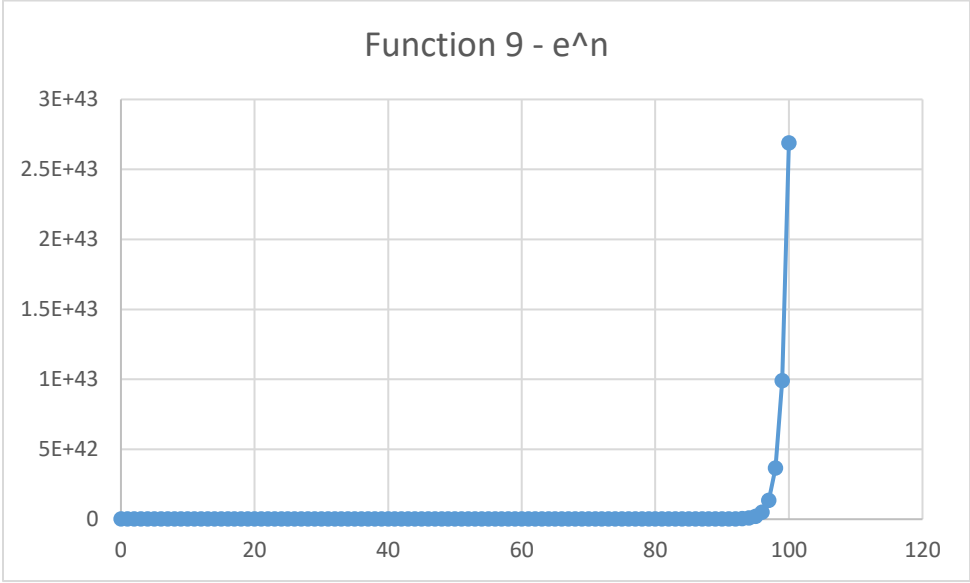
- The graph of n is a linear line.

8.



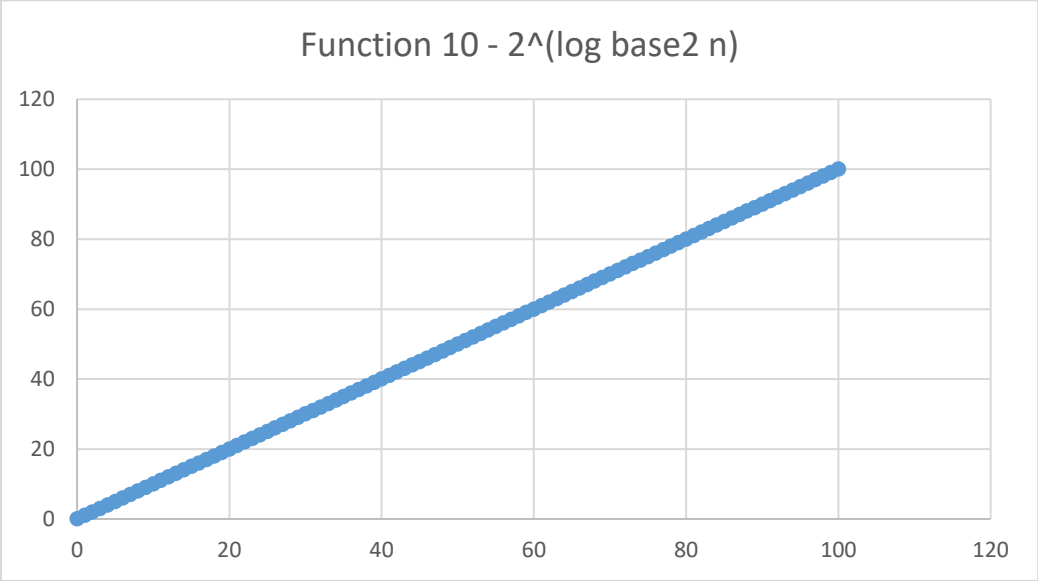
- The values for the curve exists only uptil where n = 8 after that the compiler is not able to calculate the values.

9.



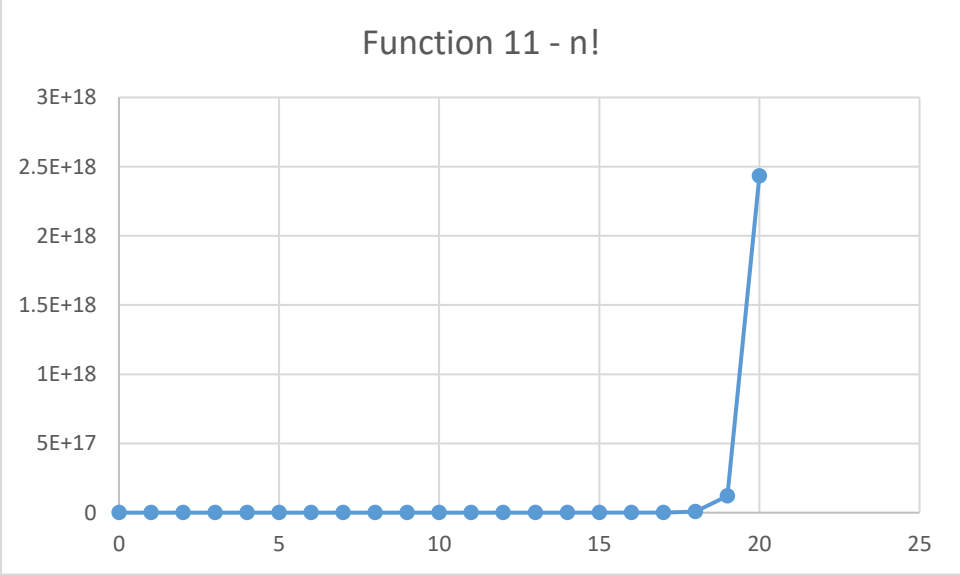
- As the function suggests, it is e^n the function is exponential, and the curve rises exponentially.

10.



- As the function on evaluating, comes out to be equal to n itself, hence the curve is linear in this case.

11.



- This is the curve of n! as mentioned in the experiment the values have been calculated upto n = 20, and here also the curve spikes up after a point as it gets multiplied by 19 and then 20.